НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«На правах рукопису»
УДК _____004.67_____

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Архітектурне рішення для обробки великих обсягів статистичних даних на пристроях з низькими технічними можливостями»

Виконав:
студент VI курсу, групи ІП-91мп
Вальчук Дмитро Володимирович _____

Науковий керівник:
старший викладач
Головченко Максим Миколайович _____

Рецензент:
Доцент кафедри ОТ, к.т.н.
Вальчук Дмитро Володимирович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Автоматизованих систем обробки інформації і управління

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма - «Інженерія програмного забезпечення комп'ютеризованих систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Вальчуку Дмитру Володимировичу

1. Тема дисертації «Архітектурне рішення для обробки великих обсягів статистичних даних на пристроях з низькими технічними можливостями», науковий керівник дисертації Головченко Максим Миколайович, старший викладач, затверджені наказом по університету від «26» жовтня 2020 р. № 3132-с

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження *аналіз даних* _____

4. Вхідні дані_____

5. Перелік завдань, які потрібно розробити *дослідження методів аналізу даних; дослідження методів відкриття файлів без великої кількості ОП; дослідження методів роботи з архівами даних; дослідження методів конвертування csv в hdf5; розробка архітектурного рішення.*_____

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____

7. Орієнтовний перелік публікацій *одні тези доповіді на науковій конференції* __

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада	Підпис, дата

	консультанта	завдання видав	завдання прийняв
Графічний	доц. Ліщук К.І.		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	*Дослідження літератури та документації*	*25.05.2020*	
2	*Огляд існуючих рішень та архітектурних підходів*	*10.07.2020*	
3	*Розробка архітектурного рішення та алгоритмів*	*20.09.2020*	
4	*Проектування системи*	*01.10.2020*	
5	*Розробка серверного та клієнтського програмного забезпечення*	*15.10.2020*	
6	*Тестування та виправлення помилок*	*01.11.2020*	
7	*Оформлення документації*	*20.11.2020*	
8	*Подання роботи на попередній захист*	*26.11.2020*	
9	*Подання роботи на основний захист*		

Студент Вальчук Дмитро Володимирович

Науковий керівник Головченко Максим Миколайович

РЕФЕРАТ

Актуальність теми. Розвиток інформаційних технологій з кожним роком займає все більше значення у багатьох галузях людської життєдіяльності, це стимулює технічний прогрес у цілому і супроводжується створенням великої кількості даних. Завдяки цьому, все з більшою швидкістю розвиваються такі напрями інформаційних технологій як машинне навчання, штучний інтелект та аналіз даних. Але не у кожної компанії є можливість найняти собі такого спеціаліста або скористатись послугами провайдерів хмарних обчислень, тому економні методи аналізу даних стають все більш цікавими для невеликих підприємств чи приватних підприємців.

Мета дослідження. Метою є побудова архітектурного рішення для аналізу великих масивів даних з мінімальним використанням оперативної пам’яті комп’ютера, а також реалізація веб додатку з використанням пропонованої архітектури.

Для реалізації поставленої мети були сформульовані **наступні завдання**:

– дослідити існуючі методи аналізу даних;

– дослідити методи швидкого завантаження файлів на сервер;

– дослідити методи роботи з архівами даних;

– дослідити методи та формати зберігання великих обсягів даних на стороні серверу;

– дослідити бібліотеки для аналізу масивів даних;

– розробити експериментальне архітектурне рішення, використовуючи Python Jupyter Notebook;

– розробити мінімальні клієнтські та серверні компоненти для підтвердження ефективності запропонованої архітектури.

Об’єкт дослідження – аналіз великих масивів статистичних даних.

Предмет дослідження – аналіз даних на пристроях з низькими технічними можливостями.

Наукова новизна: Наукова новизна полягає у розробці архітектурного рішення, яке базується на використанні жорсткого диску серверного комп’ютера

мінімізуючи при цьому використання оперативної пам’яті, що призводить до більш дешевих обчислень ніж у аналогічних рішень.

Практичне значення отриманих результатів визначається тим, що розроблено програмний продукт, який демонструє основні переваги запропонованого архітектурного рішення.

Зв’язок роботи з науковими програмами, планами, темами: дисертація виконувалась в рамках ініціативної теми кафедри АСОІУ «Методи та технології в задача пошуку та збереження даних».

Апробація: основні положення роботи доповідались і обговорювались на 5-й Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2020): Матеріали наукової конференції студентів, магістрантів та аспірантів, м. Київ, 27 листопада 2020.

Публікації:

Архітектурне рішення для обробки великих обсягів статистичних даних на пристроях з низькими технічними можливостями / Вальчук Д.В., Головченко М.М. // V Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2020): Матеріали наукової конференції студентів, магістрантів та аспірантів, м. Київ, 27 листопада 2020.

Ключові слова. АНАЛІЗ ДАНИХ, МАШИННЕ НАВЧАННЯ, ВІДОБРАЖЕННЯ ФАЙЛІВ В ПАМ’ЯТЬ, НАБІР ДАНИХ, PYTHON, VAEX, APACHE SPARK.

SUMMARY

Actuality of theme. The development of information technology is becoming increasingly important in many areas of human life, which provides technical progress in general and is accompanied by the creation of large amounts of data. Due to this, such areas of information technology as machine learning, artificial intelligence and data analysis are developing more and more rapidly. Outside of each company, the possibility of hiring such specialists or using the services of a cloud computing provider, as well as economic methods of analysis are becoming increasingly interesting for small industrial private industrial enterprises.

The aim of the research. The goal is to build an architectural solution for the analysis of large data sets with minimal use of computer RAM, as well as the implementation of a web application using the proposed architecture.

To achieve this goal, the **following tasks** were formed:

explore existing methods of data analysis;

explore methods of fast uploading files to the server;

explore methods of working with data archives;

explore methods and formats for storing large amounts of data on the server side;

develop an experimental architectural solution using Python Jupyter Notebook;

develop minimum client and server components to confirm the proposed architecture.

The object data analysis.

The subject data analysis on devices with low technical capabilities.

Scientific Novelty: The scientific novelty lies in the development of an architectural solution based on the use of the server's hard disk while minimizing the use of RAM, which leads to cheaper calculations than similar solutions.

The practical value of the results is a software product which demonstrates the main advantages of the proposed architectural solution.

Relationship with scientific programs, plans, topics: the dissertation was carried out in the scope of initiative theme of the ASOIU department “Methods and technologies in the task of searching and storing data”.

Testing: The main provisions of the work were reported and discussed at the 5th All-Ukrainian scientific-practical conference of young scientists and students "Information systems and management technologies" (ISTU-2020): Proceedings of the scientific conference of students, undergraduates and graduate students, Kyiv, November 27, 2020.

Articles:

Architectural solution for processing large volumes of statistical data on devices with low technical capabilities / Valchuk D.V., Golovchenko M.M. // V All-Ukrainian scientific-practical conference of young scientists and students "Information systems and management technologies" (ISTU-2020): Proceedings of the scientific conference of students, undergraduates and graduate students, Kyiv, November 27, 2020ю

Keywords. DATA ANALYSIS, MACHINE LEARNING, MEMORY MAPPING, DATASET, PYTHON, VAEX, APACHE SPARK.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API – (Application programming interface) – прикладний програмний інтерфейс.

DBU – одиниця обробки даних з по секундною оплатою.

DF – (DataFrame) **–** двовимірна маркована структура, яка схожа за ідеєю на звичайну таблицю, що виражається у способі її створення та роботи з її елементами.

DS – (Data set, датасет) – колекція з логічних записів, які зберігаються у вигляді кортежу. Набір даних можна порівнювати з файлом, але на відміну від файлу набір даних є одночасно і каталогом, і файлом файлової системи та не може складатися з інших наборів даних.

JIT – (Just-In-Time) – технологія збільшення продуктивності програмних систем, які використовують байт-код, шляхом компіляції байт-коду в машинний код або в інший формат безпосередньо під час роботи програми.

HDD – (Hard Disk Drive) – запам’ятовуючий пристрій довільного доступу, заснований на принципі магнітного запису.

RAM – (Random Access Memory) – енергозалежна частина системи комп’ютерної пам’яті, в якій під час роботи комп’ютера зберігається виконуваний машинний код, а також вхідні, вихідні та проміжкові дані, які оброблюються процесором.

REST – (Representational State Transfer) – архітектурний підхід до побудови веб сервісів.

SQL – (Structured Query Language) – це мова програмування структурованих запитів, яка використовується у якості ефективного засобу зберігання даних, пошуку їх частин, отримання з БД та видалення.

БД – (База даних) – це організована структура, призначена для зберігання, зміни та обробки взаємопов’язаної інформації, переважно великих розмірів.

SSD – (Solid-State Drive) – комп’ютерний енергонезалежний немеханічний запам’ятовуючий пристрій, який базується на мікросхемі пам’яті.

ВСТУП

З кожним роком об’єм даних створених людством збільшується з шаленою швидкістю. За прогнозом компанії IDC, загальний об’єм збільшиться з 33 зетабайтів у 2018 році до 175 зетабайтів у 2025 році [1]. Для більшого розуміння, один зеттабайт – це 10^{21} байтів.

За останні двадцять років швидкість створення даних зросла на два порядки, тобто за останні два роки людство згенерувало більше інформації, ніж за всю історію до цього.

Великий обсяг цієї інформації являє собою статистичні дані. IDC прогнозує, що до 2025 року шістдесят відсотків усієї інформації буде згенеровано комерційними підприємствами.

В умовах сучасної економічної кризи, розробка нових ефективних методів та алгоритмів аналізу даних набуває особливе значення для комерційних компаній та державних структур. Аналіз даних є невід’ємною частиною роботи більшості сучасних компаній [2 - 10]. Так, наприклад, випускаючи новий продукт на ринок, аналітикам потрібно знайти цільову аудиторію для цього продукту, проаналізувати вподобання потенційних користувачів та зробити висновки на яких якостях їхнього продукту робити акценти для тої чи іншої аудиторії користувачів.

Зараз існує багато методів та інфраструктур для аналізу великих обсягів даних. Основними з них є використання хмарних обчислень, так як для даного підходу немає обмежень на об’єм даних, який потрібно проаналізувати. Але цей підхід є достатньо дорогим та не всі компанії або приватні підприємці можуть собі дозволити використання цієї інфраструктури. Також, скоріш за все, звичайна людина, яка не пов’язана з програмуванням просто не буде володіти достатніми технічними знаннями для налаштування цієї інфраструктури, що породжує попит на спеціалістів у сфері аналізу даних.

З іншого боку, існують люди, які не мають можливості витрачати великі кошти на аналіз даних, але цікавляться даною сферою. Ця аудиторія хоче спробувати себе у сфері аналізу великих обсягів даних, але зазвичай не має

достатньо технічно забезпеченого комп’ютера для здійснення цієї операції. Здебільшого, аналіз великого обсягу даних на персональному комп’ютері стає неможливим через недостатність оперативної пам’яті, так як більшість сучасних методів і бібліотек відкривають файли з набором даних, завантажуючи їх у повному обсязі до оперативної пам’яті, що унеможливлює відкриття файлів, розмір яких більший за розмір цієї пам’яті.

Отже, основною метою створення архітектури було зниження вартості аналізу даних для підприємств та зацікавлених осіб.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметного середовища

Аналіз даних – важлива складова для багатьох сфер людської життєдіяльності. Тому сфера аналізу даних є дуже привабливою для молодих спеціалістів. Перш за все, люди, зацікавлені в освоєнні технологій аналізу даних, починають експериментувати на особистих комп'ютерах. Зазвичай, їх потужності вистачає для аналізу наборів даних невеликих за обсягом. Для початку цього достатньо, але з часом постає проблема аналізу даних занадто великих за обсягом. Зазвичай такі набори даних при аналізі викликають помилку з повідомленням про недостатність оперативної пам'яті для відкриття обраного набору даних.

З іншого боку, у невеликих підприємств та фізичних осіб підприємців також виникає необхідність у сфері аналізу даних. Зазвичай цій аудиторії потрібно оцінити сучасний стан тої чи іншої галузі для розуміння доцільності та попиту на їхню продукцію.

Запропоновані сьогодні методи аналізу даних є дорогими та являють собою не завжди доцільні витрати з боку малого та середнього бізнесу.

1.2 Опис процесу діяльності

Для вирішення проблеми економічної складової аналізу великих об'ємів статистичних даних було розроблено архітектуру, яка обходить обмеження оперативної пам'яті на розмір файлів, які можуть бути відкриті операційною системою шляхом використання методу memory mapping [11].

Спочатку необхідно розробити ефективну модель взаємодії користувача з запропонованою архітектурою. Для вирішення цього завдання можна створити веб застосунок з використанням клієнт-серверної архітектури [12] та спроектувати інтерфейс користувача.

Розроблений веб додаток має реалізувати авторизацію клієнтів для можливості завантаження приватних наборів даних та структуризації їх в контексті кожного користувача.

Далі необхідно дослідити методи швидкої передачі даних на сервер [13], так як очікується робота з великими обсягами статистичних даних, потрібно максимально знизити час завантаження файлів.

Наступним кроком, потрібно здійснити підготовку даних, для оптимального їх зберігання на стороні сервера. Додати можливість завантаження набору даних частинами з подальшим об’єднанням до одного файлу для подальшого додавання нових даних до завантаженого набору.

Також необхідно реалізувати агрегатні запити до наборів даних як базовий функціонал для швидкого отримання загальної статистики по обраному набору даних.

1.3 Опис функціональної моделі

Розроблена архітектура буде корисна для будь-яких користувачів, які бажають отримати статистику для заздалегідь завантаженого набору даних. Функціональними вимогами є модуль завантаження даних, модуль конвертування даних та модуль отримання статистики за обраним набором даних.

1.4 Огляд наявних аналогів

Існуючі сьогодні рішення можна поділити на дві основні групи, ті, які виконуються у локальному середовищі на комп’ютері користувача та ті, які використовують інфраструктуру хмарних обчислень [14].

Від першої групи можна виділити бібліотеку pandas [15-16] для мови програмування Python. Ця бібліотека добре підходить для аналізу невеликих наборів даних розміром до 100 мегабайтів. Вона надає високоефективні та легкі у використанні структури даних та призначена для швидкої та легкої обробки даних, читання, агрегування та візуалізації [17].

NumPy - це одна з основних бібліотек Python для обробки масивів даних. Ця бібліотека надає високопродуктивні об’єкти багатовимірних масивів та інструменти для роботи з цими масивами [18].

Бібліотека SciPy, ця бібліотека містить модулі для ефективних

математичних процедур, таких як лінійна алгебра, інтерполяція, оптимізація, інтеграція та статистика. Основний функціонал цієї бібліотеки побудований на бібліотеці NumPy та її масивах.

Якщо розглянути другу групу існуючих інструментів аналізу даних, то найбільш популярний є використання фреймворка Apache Spark [20]. Це фреймворк з відкритим вихідним кодом для розподіленої пакетної та потокової обробки неструктурованих або слабко структурованих даних, який входить в інфраструктуру Hadoop [21].

1.5 Постановка завдання

Мета

Основною метою є побудова архітектурного рішення для аналізу великих масивів даних з мінімальним використанням оперативної пам’яті комп’ютера, а також реалізація веб додатку з використанням пропонованої архітектури.

Призначення

Призначення даної роботи є створення програмного забезпечення, що виконує агрегатні запити до великого за об’ємом набору статистичних даних швидкими та економними методами.

Задачі

– дослідити існуючі методи аналізу даних;

– дослідити методи швидкого завантаження файлів на сервер;

– дослідити методи роботи з архівами даних;

– дослідити методи та формати зберігання великих обсягів даних на стороні серверу;

– дослідити бібліотеки для аналізу масивів даних;

– розробити експериментальне архітектурне рішення, використовуючи Python Jupyter Notebook;

– розробити мінімальні клієнтські та серверні компоненти для підтвердження ефективності запропонованої архітектури.

Висновки та постановка задач дослідження

Аналіз, проведений вище, показав, що існують методи та технології, які успішно використовуються у сфері аналізу даних, але вони мають багато недоліків, зокрема для розглянутих бібліотек локального аналізу даних існує обмеження у об’ємі набору даних який надається для обробки, зі збільшенням об’єму цих даних значно збільшується час на отримання результатів, а у разі, якщо розмір файлу перевищує об’єм оперативної пам’яті комп’ютера, то аналіз такого набору даних стає зовсім неможливим. Якщо розглянути технології хмарних обчислень, то проблеми з розміром вхідного файлу не виникає, але зі збільшенням об’єму цього файлу збільшується і кількість виділених ресурсів для його обробки [22], що значно впливає на вартість цих обчислень.

Отже, необхідно розробити архітектуру, яка зможе поєднати обидва підходи, тобто буде мати можливість аналізувати великі за обсягом набори даних з невеликими економічними затратами на виконання статистичних запитів, тобто архітектуру, котра не буде використовувати великої кількості оперативної пам’яті. Під час написання цього розділу було сформовано мету, призначення та задачі розробки.

2 ОБГРУНТУВАННЯ ДОСЛІДЖЕННЯ

2.1 Напрям дослідження

З першого розділу стає зрозумілим, що для розробки відповідної архітектури, необхідно вирішити багато проблем. Для початку потрібно обрати мову програмування та фреймворк для створення клієнт-серверної архітектури, після чого необхідно порівняти методи завантаження даних на сервер, обрати базу даних для структурного зберігання наборів даних на стороні сервера та обрати оптимальні методи обробки даних. Також буде наведена узагальнена структура архітектури, що розробляється.

2.2 Структура системи

Запропонована архітектура має складатися з наступних частин:

– клієнтської частини, яка буде реалізовувати інтерфейс користувача та володіти такими функціями як завантаження даних, нормалізація даних та приведення їх до необхідного для аналізу формату, створення наборів даних, додавання файлів до наборів даних, структуризація наборів даних та виконання агрегатних запитів до цих наборів даних;

– серверної частини, яка буде реалізовувати методи швидкого завантаження даних до файлової системи, конвертацію файлів та виконання агрегатних запитів.

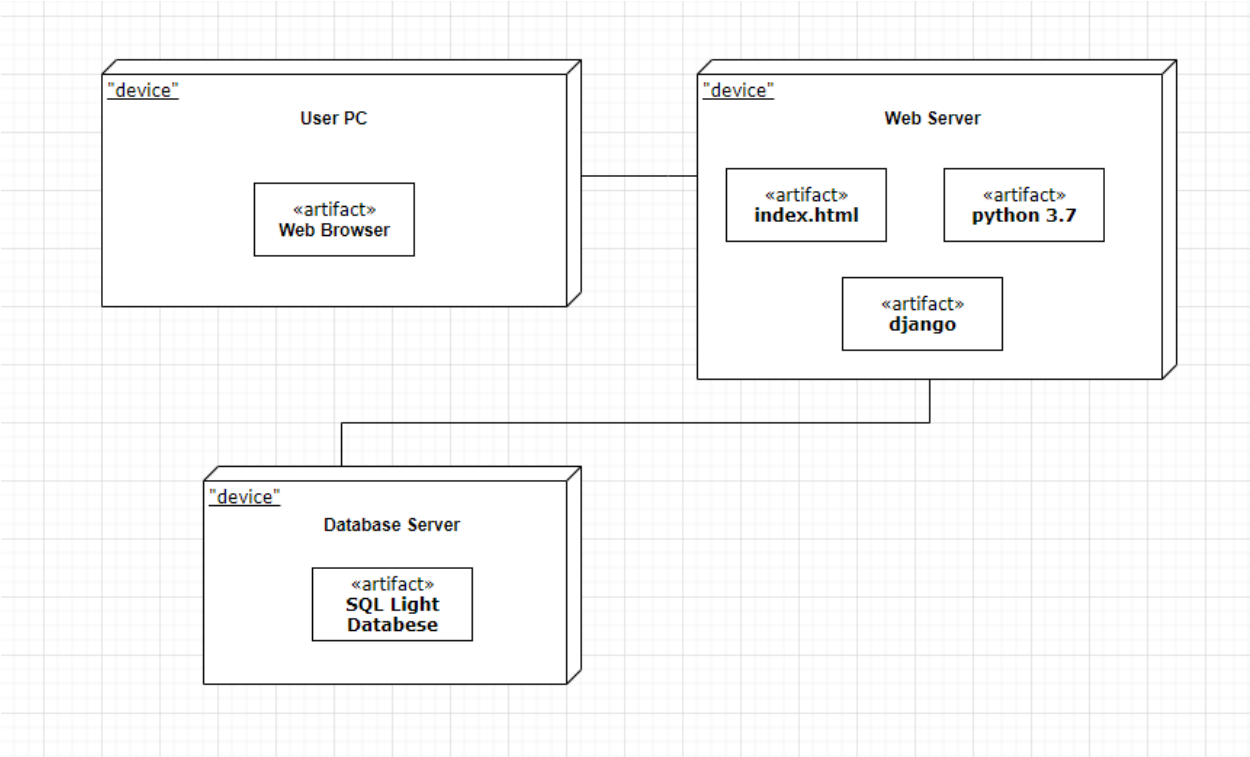


Рисунок 2.1 – Узагальнена діаграма розгортання розробленої архітектури

2.3 Методи вирішення поставлених завдань

2.3.1 Методи швидкого завантаження даних

Стандартний метод обміну даними між клієнтом та сервером – тимчасове сховище. На стороні клієнта ми виконуємо запит «записати у тимчасове сховище» та отримуємо адресу, відправляємо цю адресу серверу. Він, в свою чергу, після отримання цієї адреси виконує запит «отримати дані з тимчасового сховища». Цей метод має обмеження у вигляді максимального об’єму даних, які можна помістити у тимчасове сховище у розмірі 4 гігабайта.

Навіть, якщо б не існувало цього обмеження, в середині завантаження інтернет з’єднання може перерватися і користувачу доведеться починати завантаження файлу знову. Окрім цього, користувач навряд буде чекати таку велику кількість часу, для завантаження файлу.

Наступний метод полягає у використанні сторонніх ресурсів. В даному випадку розмір файлів необмежений, але цей підхід викликає більшу в два рази кількість операцій вводу-виводу. Перший раз дані поміщаються у так звану «хмару» і тільки на другому кроці дані надходять до серверу. Також, це накладає

додаткові витрати на використання сторонніх сервісів (провайдерів даних), а враховуючи обсяг файлів, які будуть передаватися, економічна доцільність цього підходу є невигідною.

Останній метод полягає у завантажені набору даних частинами, перш за все, це дає можливість користувачу спостерігати прогрес завантаження файлів, по друге, навіть якщо інтернет з'єднання перерветься, йому не знадобиться завантажувати цілий набір даних знову, а також це обходить обмеження у розмірі файлів в 4 гігабайта та не потребує використання сторонніх ресурсів.

2.3.2 Бібліотеки та методи аналізу даних

Pandas – це бібліотека для Python, що забезпечує швидкі, гнучкі та виразні структури даних, призначені зробити роботу з «реляційними» даними одночасно простою та інтуїтивною. Вона має на меті стати фундаментальним будівельним елементом високого рівня для практичного аналізу даних у реальному світі на Python. Ця бібліотека добре підходить для різних типів даних:

– табличні дані з неоднорідними стовбцями, як у таблиці SQL або таблиці Excel;

– впорядковані та невпорядковані дані часових рядів;

– довільні дані матриці з мітками рядків і стовпців;

– будь-яка інша форма спостережних або статистичних наборів даних.

Двома основними структурами даних бібліотеки pandas є Series та DataFrame [23]. Вони спроможні обробити переважну більшість типових завдань у таких сферах як фінансування, статистика, соціальні науки та велику кількість завдань у галузі інформаційних технологій.

Основними функціями даної бібліотеки є:

– проста обробка відсутніх даних (представлених як NaN);

– гнучкість розміру (стовпці можна додавати або видаляти з DataFrame);

– автоматичне та явне вирівнювання даних (об'єкти можуть бути явно вирівняні за набором міток, або користувач може проігнорувати мітки та дозволити DataFrame автоматично вирівнювати дані під час обчислень);

– інтуїтивне об’єднання наборів даних;

– інструменти вводу-виводу для завантаження даних з csv файлів, файлів Excel, баз даних та збереження чи завантаження файлів до формату hdf5.

Але дана бібліотека не зовсім підходить для великих обсягів даних (більших за 100 мегабайтів) через використання оперативної пам’яті комп’ютера.

Dask – це бібліотека для Python, яка підтримує DataFrame бібліотеки pandas і структури даних (масиви) Numpy [24]. Цю бібліотеку можна запускати як на локальному комп’ютері, так і масштабувати і запускати у кластері. Загалом, код пишеться один раз, а потім можна обрати, запускати його у локальному середовищі чи розгорнути у кластері з безлічі вузлів, використовуючи при цьому звичайний синтаксис Python. Найбільшою перевагою використання даної бібліотеки є можливість з мінімальними змінами у коді розпаралелювати його, використовуючи вже існуючі обчислювальні потужності. При паралельній обробці даних програма виконується швидше та залишається більше часу на аналітику.

Apache Spark – це платформа паралельної обробки з відкритим кодом, яка підтримує обробку в пам’яті для того, щоб підвищити продуктивність додатків, які аналізують великі обсяги даних. Це рішення створено для роботи з великими обсягами даних та використовується для обробки даних з дуже великим об’ємом або складністю для традиційних баз даних.

Дана архітектура корисна, якщо є необхідність зберігати або обробляти великі обсяги даних, перетворювати неструктуровані дані або обробляти потокові дані. Spark надає механізм розподіленої обробки широкого призначення, який дозволяє реалізувати декілька сценаріїв роботи з великими даними. До основних функцій даної архітектури можна віднести:

– обробка потоків даних в реальному часі;

– пакетна обробка;

– машинне навчання з використанням MLlib;

– обробка графів за допомогою GraphX;

– обробка SQL та структурування даних за допомогою Spark SQL.

Для Apache Spark при використанні архітектури «основний-робочий», передбачено три основних компоненти: драйвер, виконавець та диспетчер кластеру.

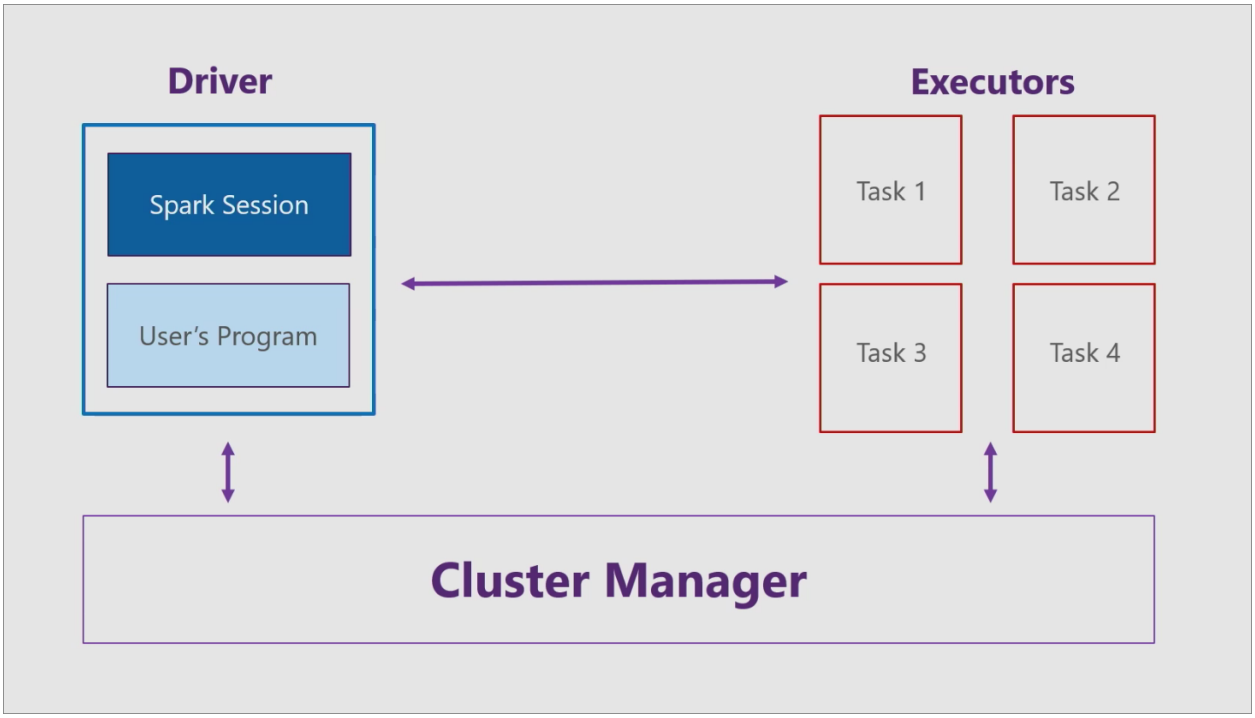


Рисунок 2.2 – Архітектура Apache Spark «основний-робочий»

Vaex – це потужна Python-бібліотека, призначена для організації «лінивої» обробки DataFrame, схожими на pandas DataFrame, розрахована на візуалізацію та дослідження великих наборів табличних даних. Ця бібліотека може обчислювати основні статистичні змінні для досліджуваних даних. При цьому обробка мільярда записів за час рівний одній секунді. Бібліотека підтримує багато засобів візуалізації даних, що допомагає досліджувати великі обсяги даних у інтерактивному режимі.

Бібліотека Vaex не схожа на Dask, але структури, які в ній використовуються схожі на DataFrame. Вони побудовані на базі датафреймів pandas. Це означає, що Dask отримав у спадок від pandas певні обмеження для роботи з датафреймами. Наприклад, дані, які плануються для обробки, повинні бути повністю завантажені в оперативну пам'ять [25].

Vaex не робить копій датафреймів, в результаті чого, за допомогою даної бібліотеки можна опрацьовувати великі датафрейми на пристроях з невеликим

об’ємом оперативної пам’яті. Як Vaex так і Dask використовують «ліниві» методи обробки даних. Основні відмінності між ними полягають у тому, що Vaex вираховує значення полів лише у тому випадку, коли вони потрібні, а в Dask потрібно явним способом використовувати функцію обчислення значень.

Основні можливості бібліотеки Vaex:

– використання віртуальних стовпців – при додаванні до набору даних нового стовпця, створюється віртуальний стовпець, який не займає пам’яті, а значення які в ньому зберігаються, обчислюються «на льоту». Наприклад, для виразу np.sqrt(ds.x**2 + ds.y**2) обчислень не відбудеться;

– ефективна фільтрація даних – при фільтрації даних, копії датафреймів не створюються, це сприяє більш ефективному використанню пам’яті;

– агрегування даних – при агрегуванні даних Vaex комбінує операції групування і агрегації даних в одну команду;

– об’єднання даних – при об’єднанні даних не створюються копії цих даних в пам’яті, це сприяє економії використання пам’яті.

Багато вищезазначених методів мають певні обмеження, але вони достатньо легко усуваються за допомогою використання додаткових бібліотек. Так, наприклад, обчислення виразу np.sqrt(ds.x**2 + ds.y**2) на льоту може зайняти певний час. Для вирішення цього завдання ми можемо використати такі бібліотеки як Pythran чи Numba, що допоможе оптимізувати обчислення за рахунок використання Just-In-Time компіляції.

```
df['r_normal'] = np.sqrt(df.x**2 + df.y**2)
df['r_jit'] = np.sqrt(df.x**2 + df.y**2).jit_numba()
```

```
%%timeit
df.mean(df.r_normal)
```

```
34.1 ms ± 4.08 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
%%timeit
df.mean(df.r_jit)
```

```
14 ms ± 124 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Рисунок 2.2 – Підвищення швидкості виконання запитів за допомогою використання Numba

Також, JIT компіляція виразів підтримується для датафреймів, які знаходяться на стороні сервера (компіляція відбувається на стороні сервера)[26]. Якщо технічний пристрій, на якому здійснюються обчислення, володіє достатнім обсягом оперативної пам’яті, то існує метод для матеріалізації віртуального стовпця, що дає змогу отримати додатковий обсяг обчислювальних потужностей за рахунок використання RAM.

```
df['r_normal'] = np.sqrt(df.x**2 + df.y**2)
df['r_jit'] = np.sqrt(df.x**2 + df.y**2).jit_numba()
```

```
%%timeit
df.mean(df.r_normal)
```

```
34.1 ms ± 4.08 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
%%timeit
df.mean(df.r_jit)
```

```
14 ms ± 124 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Рисунок 2.3 – Приклад матеріалізації віртуального стовпця

2.4 Фреймворки для створення клієнт-серверної архітектури.

Сьогодні, найпопулярнішою мовою для аналізу даних є Python. Більшість знайдених методів аналізу статистичних даних були реалізовані за допомогою цієї мови програмування, тому доцільно використовувати її і для написання клієнт-серверної архітектури. Проаналізувавши існуючі фреймворки, було вирішено виділити основні з них.

Django є найбільш використовуваним фреймворком для Python [27]. У нього доволі широкий функціонал, це дозволяє зосередитися на розробці, не витрачаючи часу на організацію структури проекту.

Особливості:

- об’єктно-реляційне перетворення;
- маршрутизація URL-адрес та їх представлень;
- механізм шаблонів;
- форми;
- ідентифікація;
- адміністративна панель;
- інтернаціоналізація;
- безпека.

Flask – мікрофреймворк Python, який має модульний дизайн [28]. Цей фреймворк призначений для створення веб-додатків. Також, він не має визначеної системи баз даних або системи ORM, для використання баз даних знадобиться завантажувати окремий модуль.

Особливості:

- маршрутизація URL-адрес та їх представлень;
- механізм шаблонів;
- керування сеансами.

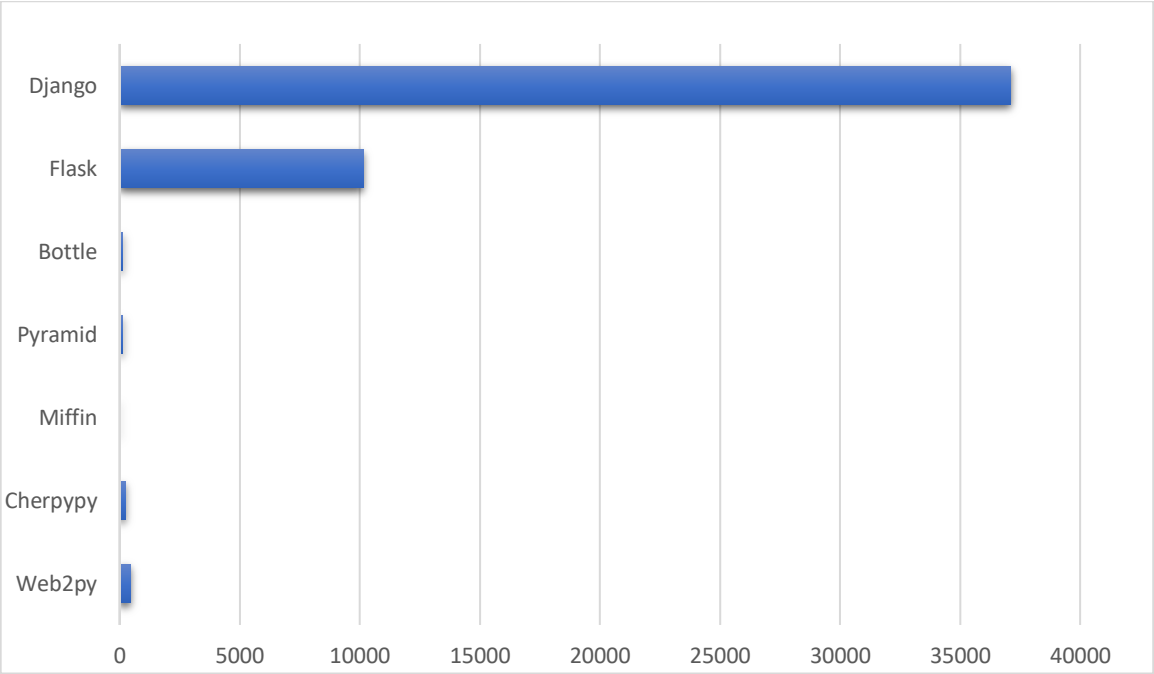


Рисунок 2.4 – Популярність фреймворків на GitHub

Висновки до розділу

Отже, враховуючи інформацію надану у цьому розділі, можна зробити висновки, що використання бібліотеки Vaex у якості двигуна для аналізу даних є вигідним, так як дана бібліотека може оперувати великими обсягами статистичних даних не завантажуючи їх до RAM. Вона відіграє вирішальну роль у економічній складовій запропонованого архітектурного рішення.

Далі, з існуючих фрейморків для створення клієнт-серверної архітектури на мові програмування Python було вирішено обрати Django. Вирішальною перевагою цього фреймворку стала його популярність, що дає змогу передбачити подальший розвиток його функціоналу та велику спільноту користувачів.

3 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

Розробка архітектури буде відбуватися на мові програмування Python. Для розробки клієнт-серверної архітектури було прийнято рішення обрати фреймворк Django. У якості бази даних для структурування користувацьких файлів використовується SQLITE, так як має найпростішу інтеграцію у проекти Django і достатньо функціональності для покриття необхідних задач. Front-end частина проекту реалізована на так званих шаблонах Django, що пришвидшує процес написання коду та дає більш гнучкий функціонал ніж використання статичних веб-сторінок.

3.1 Налаштування проекту Django

Розробка почалася з встановлення інтерпретатора Python та пакетного менеджера pip, одразу після чого можна приступати до завантаження та налаштування проекту Django.

На першому кроці необхідно завантажити та встановити віртуальне середовище, для того, щоб програма була незалежною від інших програм розроблених на Python. Якщо не створювати віртуальне середовище, то всі пакети для Python будуть встановлюватися глобально, через що, при оновлені пакетів для інших проектів нам доведеться оновлювати код розробленої програми через зворотну залежність цих пакетів.

Після встановлення віртуального середовища необхідно його активувати. Активація відбувається шляхом запуску файлу activate.bat, який знаходиться у папці Scripts створеного віртуального середовища.

Отже, після активації віртуального середовища, нам необхідно встановити пакет Django. Далі в проекті буде створено структуру файлів та папок для подальшої розробки. Також, буде автоматично створений файл бази даних db.sqlite3. Для перегляду таблиць та даних створеної БД можна використати менеджер баз даних DB Browser for SQLite.

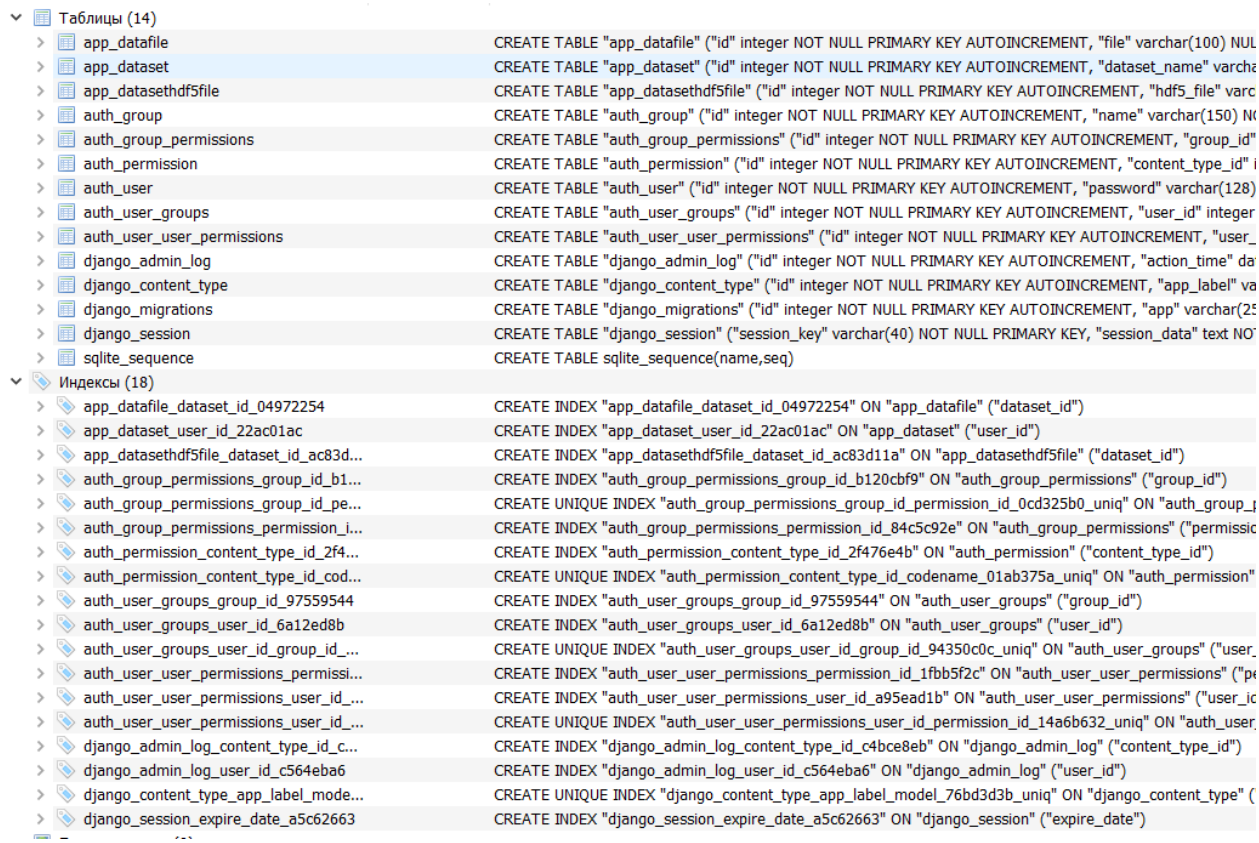


Рисунок 3.1 – Згенеровані фреймворком таблиці

Для спрощення процесу розробки було вирішено не створювати новий модуль авторизації користувачів, а взяти згенеровану структуру. За допомогою даного підходу зникає необхідність в розробці таких модулів як логін, адміністративна панель та значно простіше працювати з контекстом користувача під час запитів на сервер.

3.2 Визначення формату вхідних файлів

Більшість наборів даних являють собою файли формату .csv. Вони можуть досягати дуже великого об’єму в залежності від розміру даних, які зберігаються у обраному наборі даних. Для можливості обробки цих даних, їх необхідно передати на сервер. Якщо спробувати передавати файл набору даних через один запит до серверу, то одразу виникає декілька слабких місць у цьому підході.

Перше – це те, що у користувача може перерватися інтернет з’єднання під час передачі цих даних, що призведе до необхідності повторного завантаження файлу. Зважаючи на те, що розміри файлів зазвичай мають великий об’єм, час передачі даних на сервер буде більшим, за декілька хвилин, що значно підвищує

вірогідність втрати інтернет зв’язку.

Друге – це те, що через деякий час після завантаження даних на сервер, у набір даних можуть бути додані нові статистичні записи і користувачу доведеться видаляти вже завантажений на сервер набір даних і знову перезавантажувати оновлений набір даних у повному обсязі.

Було розглянуто два варіанти вирішення, для вирішення вищезазначених проблем. Перший – приймати у вигляді вхідного файлу заархівований у форматі .zip набір даних. Перевагами цього формату, наприклад, над форматом .rar є:

– швидкість архівування даних більша, що покращує користувацький досвід;

– на відміну від RAR, має можливість створення «неперервного архіву», що надає змогу працювати з ним, використовуючи бібліотеку zipfile без розробки і витрати процесорного часу на конвертацію архіву RAR у неперервний;

Другий – приймати на вхід набір даних у вигляді файлів частин оригінального набору, що дасть змогу легко оперувати певними частинами набору даних (додавати/видаляти), а також призведе до зменшення вірогідності втрати інтернет з’єднання користувачем.

Обидва представлені підходи вирішують виявлену проблему, але найбільш оптимальний варіант доведеться досягти об’єднавши їх разом.

В результаті, на вхід користувач повинен завантажити частини набору даних заархівованих у формат .zip. Після отримання даних сервером, виконати функцію вилучення даних, під час якої кожна частина набору даних буде вилучена до папки, після чого автоматично запуститься метод збору файлів у цій частині до єдиного файлу .csv та цей файл буде доданий до таблиці датасету. Також, під час виконання функції об’єднання файлів необхідно пам’ятати, що дані з них копіюються до оперативної пам’яті комп’ютера, та, враховуючи очікуваний вхідний розмір файлів, можуть призвести до помилки «MemoryError», тобто недостатню кількість оперативної пам’яті. Для вирішення цієї проблеми можна встановити параметр chanksize для файлу, що

відкривається, це призведе до того, що вказану кількість байтів у цьому параметрі буде зчитано та записано у вихідний файл, після чого ці дані будуть вилучені із RAM.

3.3 Створення моделей даних

Для структурування даних для конкретного користувача постало завдання розробити модель даних для зберігання даних датасету. У системі було передбачено наступні таблиці для зберігання даних (Рис. 3.2).

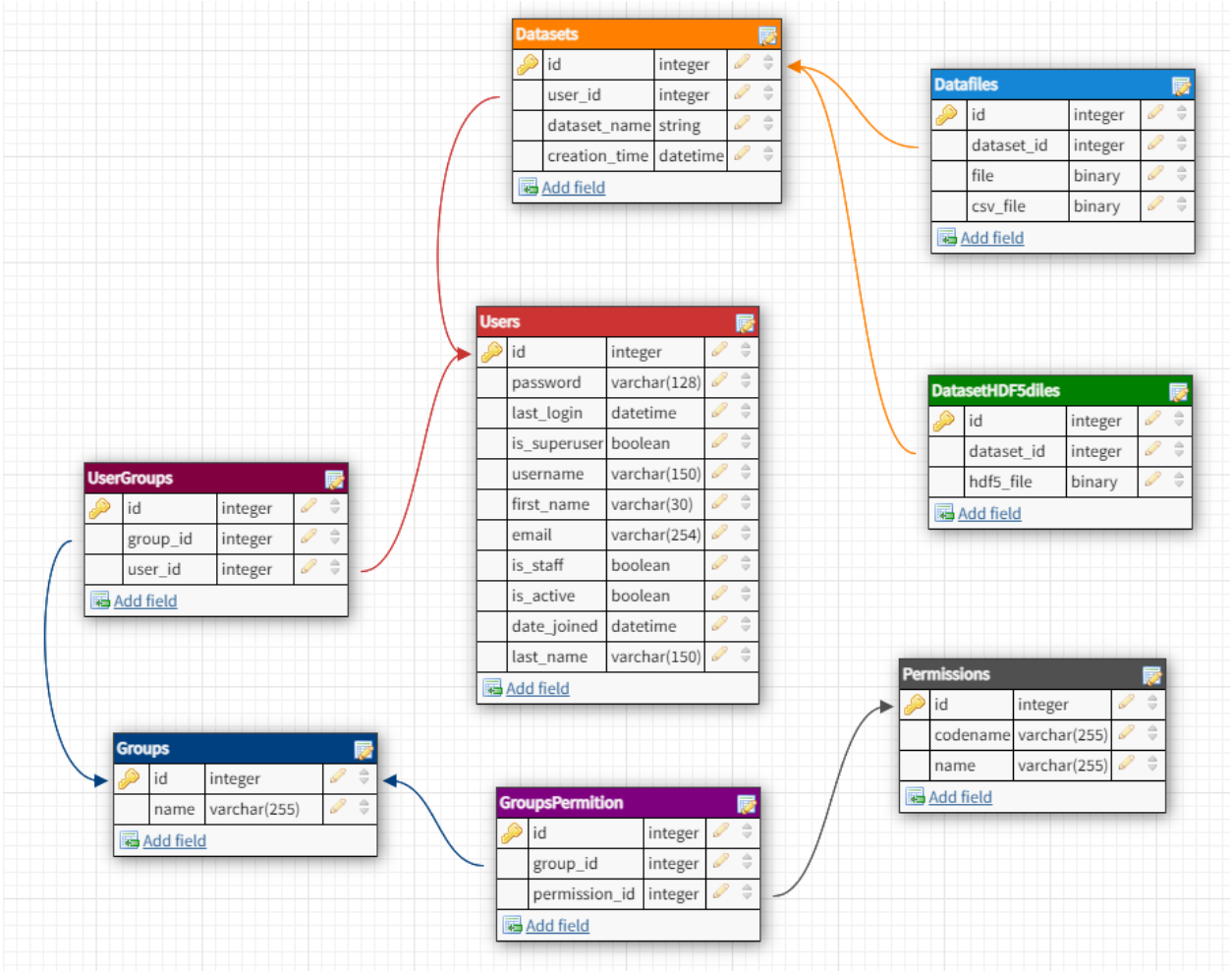


Рисунок 3.2 – Схема основних таблиць бази даних

Нижче представлено опис логічної структури таблиць (таблиця 2.1).

Таблиця 2.1 – Опис логічної структури БД

Назва таблиці	Призначення
Users	Таблиця користувачів
Groups	Таблиця користувацьких груп

Продовження таблиці 2.1

Назва таблиці	Призначення
UserGroups	Таблиця відношення користувачів до груп
Permissions	Таблиця системних дозволів
GroupsPermition	Таблиця відношення дозволів до груп
Dataset	Таблиця наборів даних користувачів
Datafile	Таблиця файлів набору даних.
DatasetHDF5File	Таблиця файлу набору даних, який готовий до аналізу

Далі, таблиці бази даних представлені у табличному форматі: ім’я таблиці, назва, тип даних та інформація про стовбець (таблиця 2.2).

Таблиця 2.2 – Детальний опис структури таблиці

Назва таблиці	Назва стовпця	Тип даних	Інформація
Users	Id	Int	Ідентифікатор користувача
	Password	Nvarchar(128)	Пароль користувача
	Last_login	DateTime	Дата та час останнього входу у систему
	Is_superuser	Boolean	Чи є користувач модератором
	Username	Nvarchar(150)	Логін користувача
	First_name	Nvarchar(30)	Ім’я користувача
	Email	Nvarchar(24)	Поштова адреса користувача
	Is_active	Boolean	Чи є користувач активний у даний час
	Date_joined	DateTime	Дата реєстрації
	Last_name	Nvarchar(150)	Прізвище користувача

Продовження таблиці 2.2

Назва таблиці	Назва стовпця	Тип даних	Інформація
Groups	Id	Int	Ідентифікатор групи
	Name	Nvarchar(255)	Назва групи
UserGroups	Id	Int	Ідентифікатор запису користувач-група
	Group_id	Int	Ідентифікатор групи
	User_id	Int	Ідентифікатор користувача
Permissions	Id	Int	Ідентифікатор дозволу
	Codename	Nvarchar(255)	Код дозволу
	Name	Nvarchar(255)	Ім’я дозволу
GroupsPermition	Id	Int	Ідентифікатор запису група-дозвіл
	Group_id	Int	Ідентифікатор групи
	Permission_id	Int	Ідентифікатор дозволу
Dataset	Id	Int	Ідентифікатор набору даних
	User_id	Int	Ідентифікатор користувача
	Dataset_name	Nvarchar(255)	Назва набору даних
	Creation_time	DateTime	Час створення набору
Datafiles	Id	Int	Ідентифікатор файлу даних
	Dataset_id	Int	Ідентифікатор набору даних, до якого відноситься файл
	File	Binary	Файл даних у форматі .zip
	Csv_file	Binary	Файл даних у форматі .csv

Продовження таблиці 2.2

Назва таблиці	Назва стовпця	Тип даних	Інформація
DatasetHDF5File	Id	Int	Ідентифікатор групи файлів набору даних
	Dataset_id	Int	Ідентифікатор набору даних до якого відносяться файли
	Hdf5_file	Binary	Файл набору даних у форматі .hd5
	Csv_file	Binary	Файл набору даних у форматі .csv

3.4 Модуль аналізу даних

При розробці даної архітектури було вирішено обрати бібліотеку vaex для здійснення запитів для аналізу набору даних. Вона продемонструвала гарні показники швидкості відкриття файлів та виконання запитів до них. Вона має можливість обчислювати статистичні дані, такі як середнє значення, сума, кількість та стандартне відхилення, на n-мірній сітці до мільярду (10^9) об’єктів/рядків в секунду. Візуалізація даних виконується з використанням гістограм, графіків щільності та тривимірного об’ємного рендерингу, що дозволяє інтерактивно досліджувати велику кількість даних. Vaex використовує метод відображення пам’яті, політику нульового копіювання та ліниві обчислення для покращення продуктивності системи без витрат RAM.

Бібліотека розділена не декілька компонентів:

– vaex-core – DataFrame та основні алгоритми, приймає декілька масивів у якості вхідних стовпців;

– vaex-hdf5 – провідник масивів numpy з методами відображення пам’яті в датафреймах;

– vaex-arrow – провайдер для обміну даними між мовами;

– vaex-viz – візуалізація даних на базі бібліотеки matplotlib;

– vaex-jupyter – інтерактивна візуалізація на основі віджетів ipywidgets Jupyter, bqplot, ipyvolume та ipyleaflet;

– vaex-astro – перетворення, пов’язані з астрономією, та підтримка файлів FITS.

– vaex-server – надає сервер для віддаленого доступу до датафреймів;

– vaex – мета-пакет, який встановлює усе вищеперераховане.

3.5 Модуль попередньої обробки даних.

Для попередньої обробки наборів даних були розроблені наступні методи:

3.1.1 Завантаження набору даних.

Даний метод отримує назву набору даних та його частини у вигляді заархівованих файлів. За один раз можна завантажити необмежену кількість цих частин. Потім перевіряє чи існує набір даних з такою ж назвою у контексті користувача. В залежності від результату, даний метод створює новий запис у таблиці наборів даних або підв’язує файли до існуючого набору, додаючи їх у таблицю файлів з ідентифікатором датасету. Після цього, метод повертає користувача на головну сторінку, на якій відображаються список його наборів та файлів даних.

3.1.2 Видалення наборів та файлів даних

Цей функціонал дає можливість користувачу видаляти набори даних у повному обсязі, або лише деякі файли з нього. На вхід цих методів приймаються ідентифікатори набору або файлу, який користувач бажає видалити.

3.1.3 Розархівація набору даних.

У даному методі відбувається вилучення даних з архівів за допомогою бібліотеки zipfile. Дані вилучаються у теку з назвою архіву і об’єднуються у єдиний .csv файл. Для запобігання використання великої кількості оперативної пам’яті було встановлено chunksize у розмірі 50000 байтів. Після об’єднання файлів та зберігання створеного файлу у папці контенту проекту, створена папка з частинами наборів даних видаляється з метою економії місця на жорсткому диску віддаленого сервера.

При розархівації повного набору даних, може бути витрачена значна кількість часу, тому для зручності було розроблено функцію розархівації

конкретної частини набору даних.

Висновки до розділу

У даному розділі було детально описано процес розробки програмного коду запропонованої архітектури. Також були наведені деякі перешкоди, без вирішення яких, кінцевий продукт міг бути незручним у використанні.

Враховуючи специфіку сфери використання запропонованої архітектури, була передбачена можливість зручної та швидкої праці з великими обсягами файлів даних і була передбачена можливість керування завантаженими наборами даних користувачем.

4 ТЕСТУВАННЯ АРХІТЕКТУРИ

4.1 Опис веб додатку

При запуску додатку вперше, до бази додаються дані адміністратора, з якими можливо авторизуватись у веб застосунку. Користувачам доступна лише сторінка авторизації адміністратора (рис. 4.1), реєструвати користувачів у системі має право лише адміністратор системи.

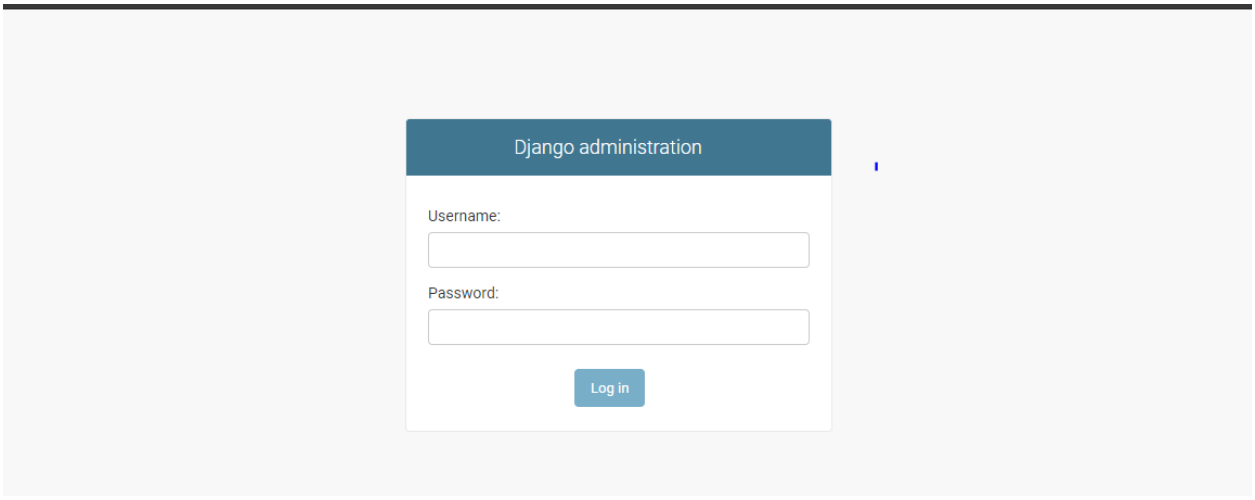


Рисунок 4.1 – Сторінка авторизації адміністратора

Після входу у систему, адміністратор може створити групу для користувачів або додати користувача, а також переглянути останні дії.

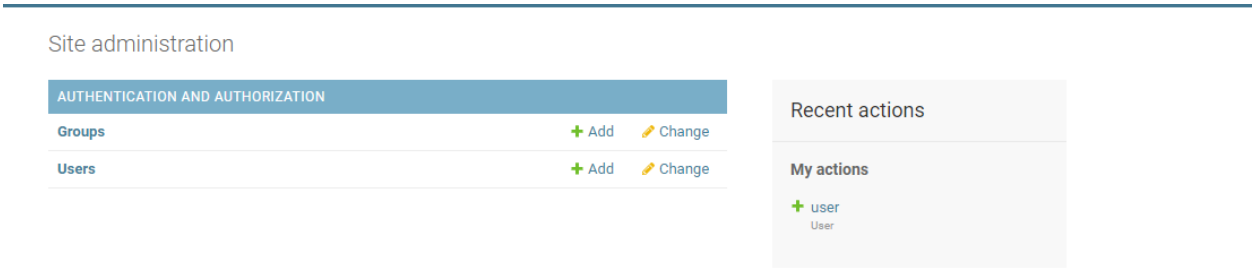


Рисунок 4.2 – Головна сторінка адміністративної панелі

Також, під час першого запуску, створюються дві групи користувачів, це адміністратори та звичайні користувачі. За бажанням адміністратор може створити власну групу користувачів (Рис. 4.3) та надавати ним різні права доступу до додатку. При створені групи користувачів адміністратор вказує назву групи та права доступу до компонентів та методів додатку.

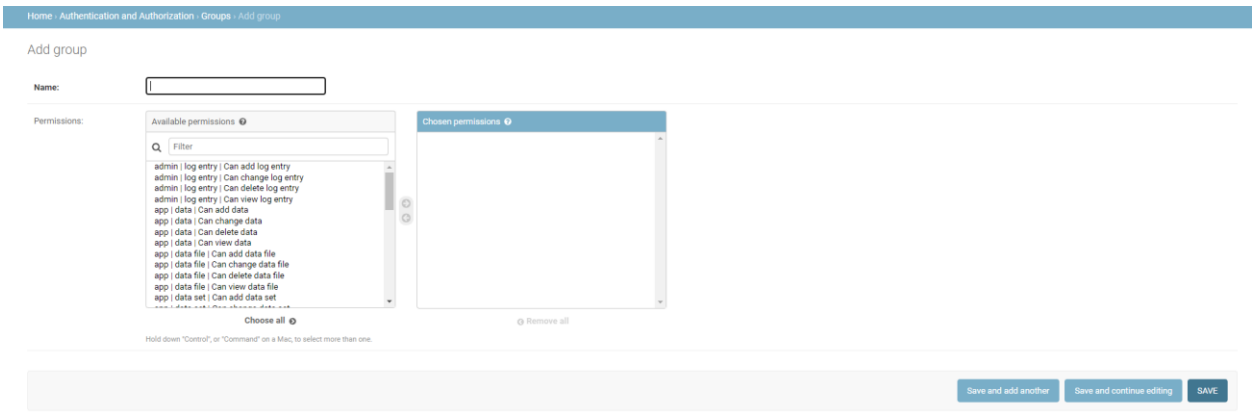


Рисунок 4.3 – Сторінка створення групи користувачів з певними правами

Якщо, необхідна для адміністратора група вже існує, він може створити користувача (Рис. 4.4). При створені користувача адміністратору необхідно вказати його логін, пароль та підтвердити пароль та, за необхідності, вказати групу, до якої буде належати користувач.

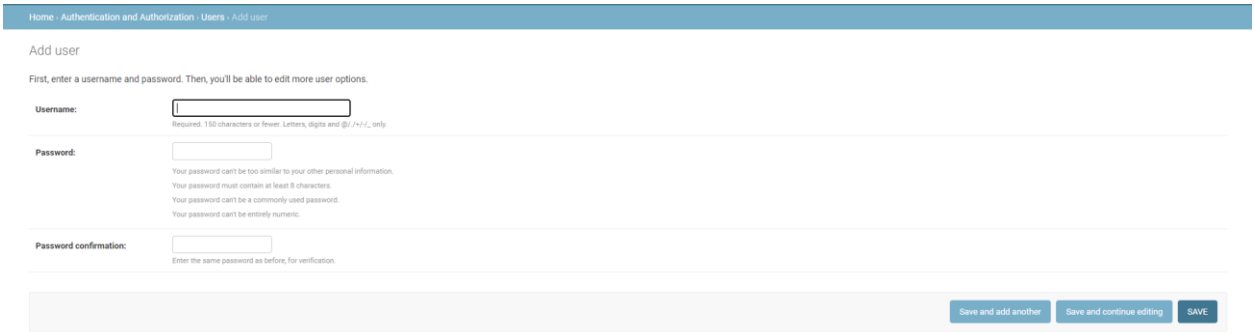


Рисунок 4.4 – Сторінка створення користувача

Після створення нового користувача, адміністратор має можливість переглянути його інформацію на сторінці списку користувачів (Рис. 4.5). На цій сторінці він може відредагувати або видалити користувача, а також скористатися функціями фільтрації користувачів за різними параметрами для більш зручного пошуку.

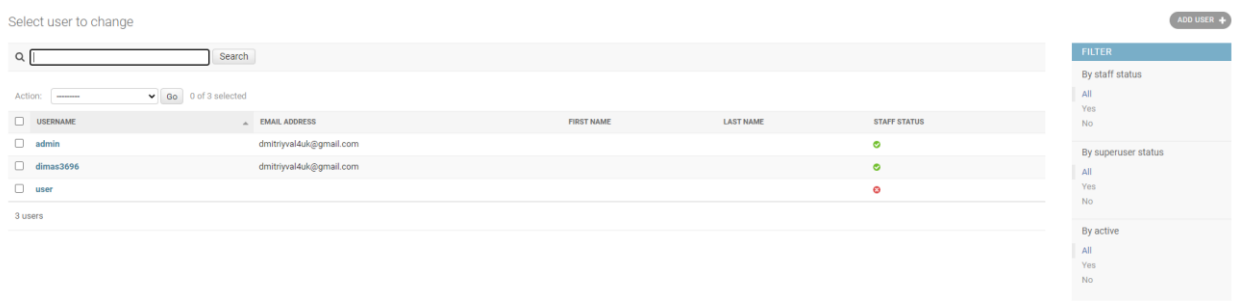


Рисунок 4.5 – Сторінка списку користувачів

При переході на сторінку редагування користувача (Рис. 4.6), адміністратор має можливість змінити його логін та переглянути хеш паролю. Також він може додати персональну інформацію для цього користувача, таку як ім’я, прізвище та адреса електронної пошти. Окрім цих методів, адміністратор може змінити права доступу користувача.

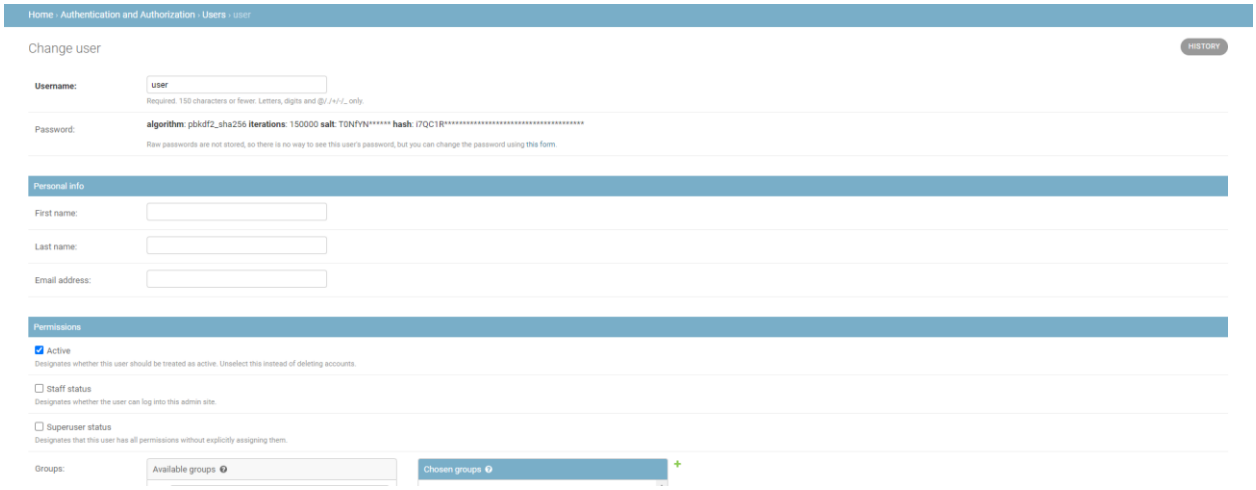


Рисунок 4.6 – Сторінка редагування користувача

Після створення адміністратором користувача, останній має можливість перейти до додатку. Якщо користувач не авторизований, його автоматично перенаправить на сторінку авторизації користувача (Рис. 4.7).

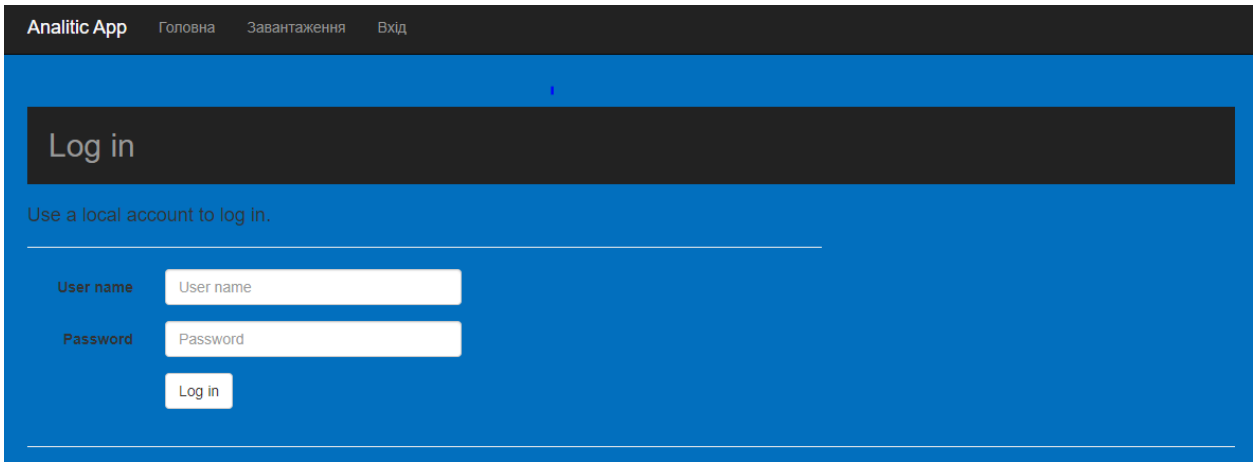


Рисунок 4.7 – Сторінка авторизації користувача

При успішному проходженні авторизації користувачем, йому відкриється головна сторінка додатку на якій буде відображено список створених ним наборів даних та файлів у них в табличному вигляді (Рис. 4.8). Після авторизації у користувача, замість посилання Login, з’являється привітання з ім’ям користувача та можливість вийти з сайту (змінити користувача).

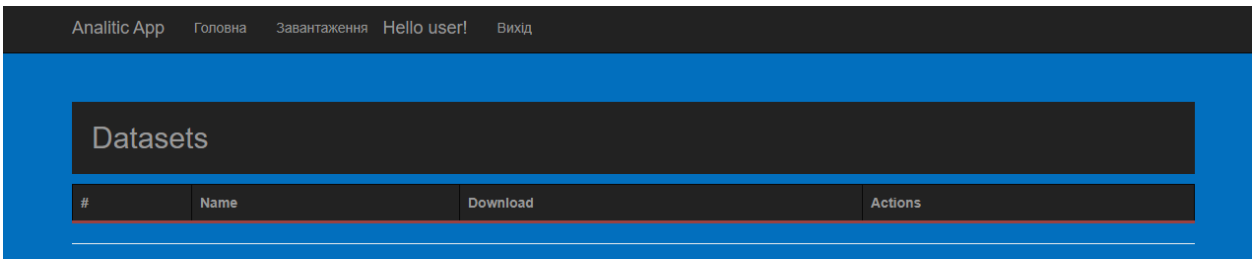


Рисунок 4.8 – Головна сторінка додатку для користувача

Далі, для користувача стає доступне посилання для переходу на сторінку завантаження набору даних (Рис. 4.9). Якщо у користувача вже існує хоча б один створений набір даних, система автоматично підтягне його ім’я у поле назву датасету. Якщо користувач хоче довантажити частини набору даних, вони автоматично будуть додані до набору з таким самим ім’ям та створеного цим користувачем. Після завантаження файлів користувача буде повернуто на головну сторінку, на якій уже буде відображений створений набір даних та додані до нього файли.

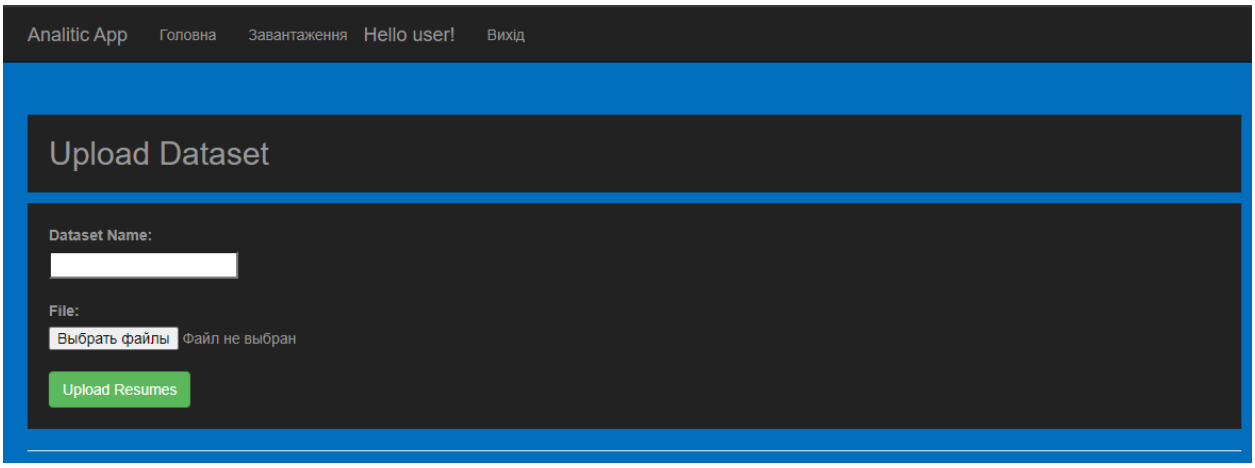


Рисунок 4.9 – Сторінка завантаження набору даних

На головній сторінці, після створення набору даних та додавання да нього файлів для користувача відкриються декілька методів (Рис. 4.10).

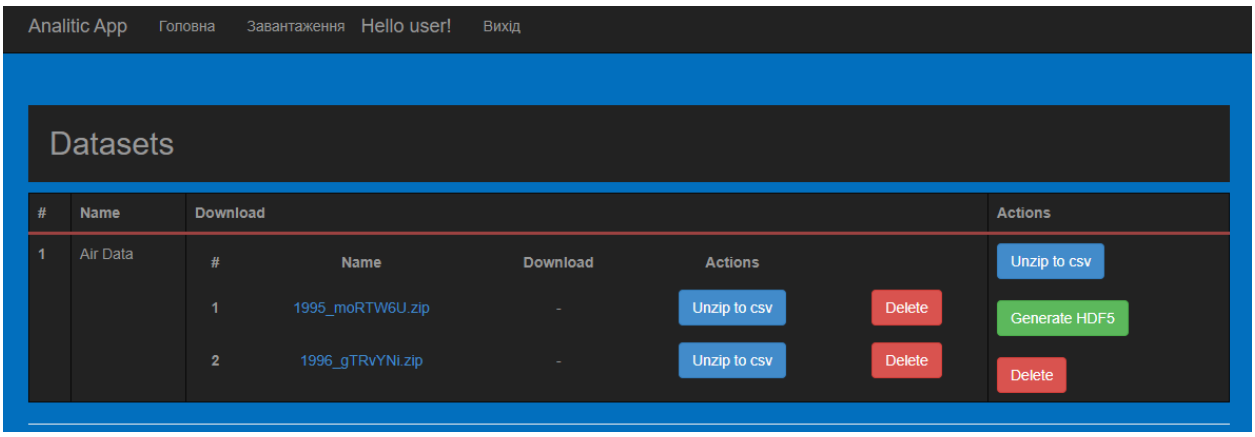


Рисунок 4.10 – Таблиця з набором даних, файлів у ньому та методами обробки

При натисканні на назву файлу у наборі даних, файл буде завантажений на локальний комп’ютер користувача (Рис. 4.11).

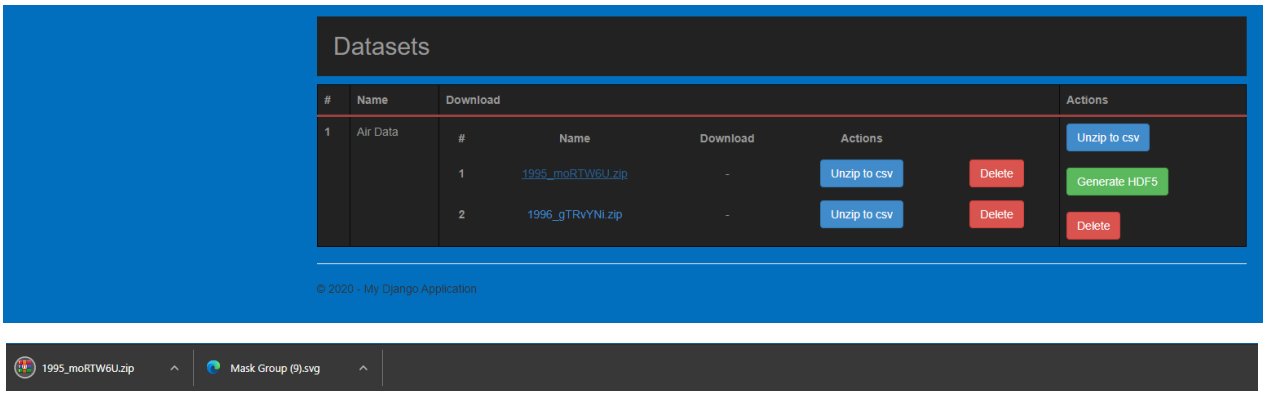


Рисунок 4.11 – Завантаження набору даних у форматі .zip

При натисканні на кнопку «Unzip to csv» у контексті файлу завантажений файл буде розархівовано в папку з ім’ям файлу (Рис. 4.12). Після чого всі файли із створеної папки будуть об’єднані до єдиного файлу «output.csv» і додані у таблицю файлів набору даних (Рис. 4.13), відображена на головній сторінці у таблиці набору даних (Рис. 4.14) та часткові файли будуть видалені (Рис. 4.15). Аналогічний метод можна застосувати у контексті набору даних, він призведе до послідовного перетворення усіх файлів в обраному наборі.

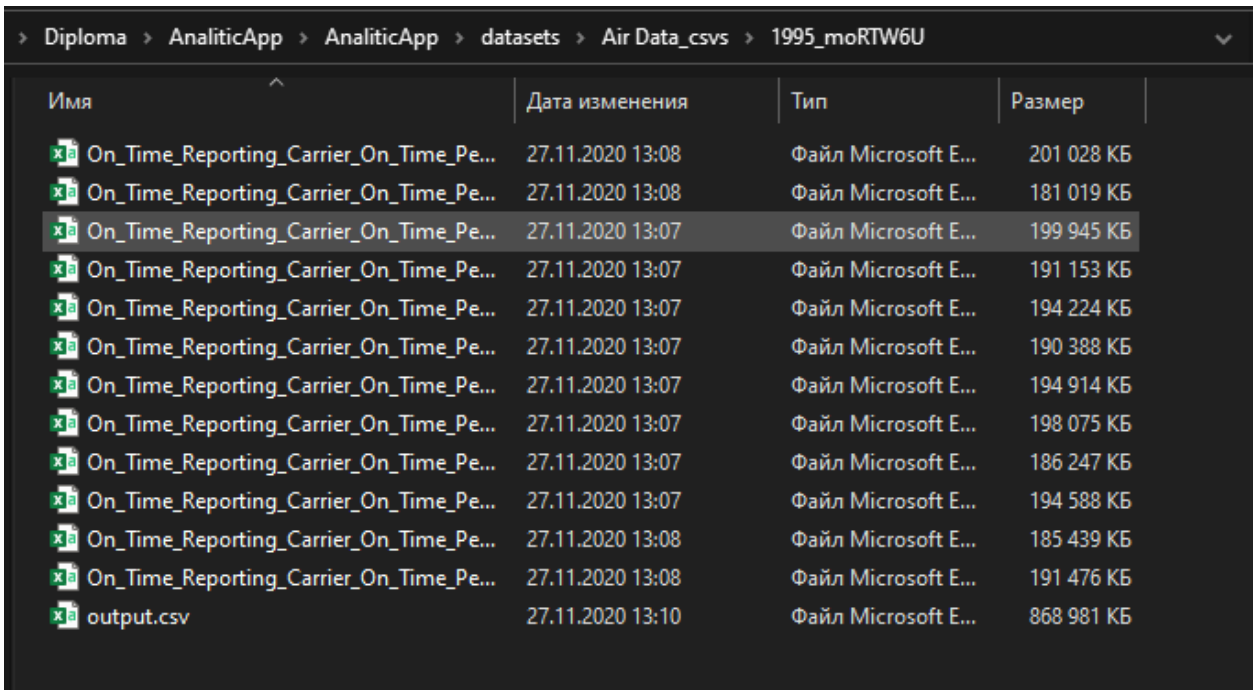


Рисунок 4.12 – Папка розархівованого файлу набору даних

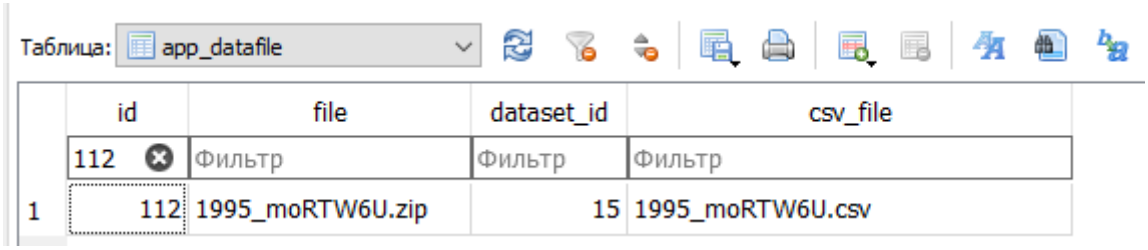


Рисунок 4.13 – Таблиця файлів наборів даних

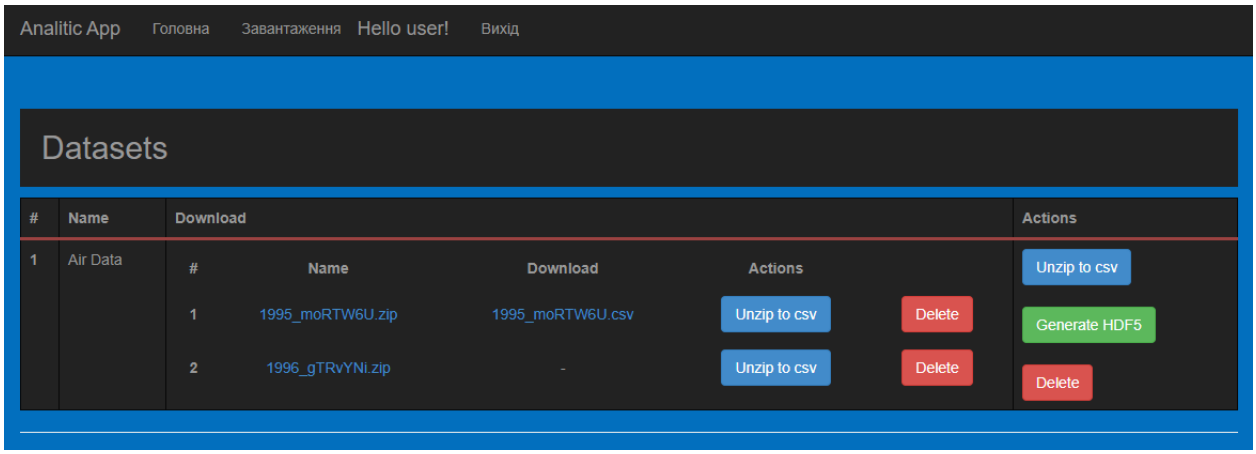


Рисунок 4.14 – Таблиця набору даних користувача з доданим файлом даних

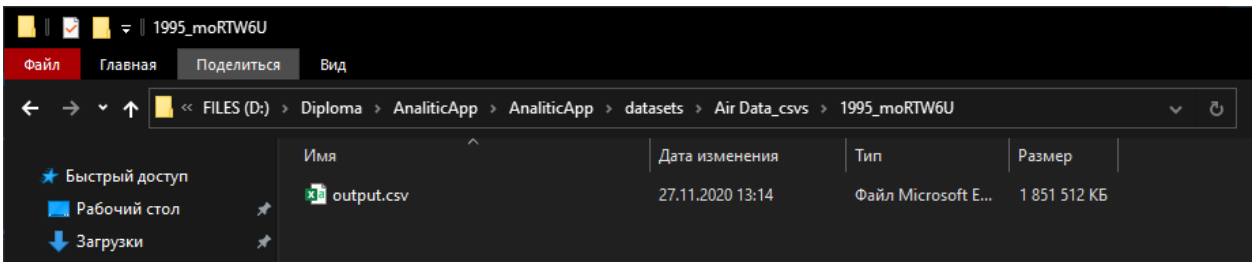


Рисунок 4.15 – Очищена папка з об’єднаним файлом

При натисканні на кнопку «Generate HDF5» у контексті частини набору даних файл архіву буде перетворено у вихідний файл з ім’ям частини набору даних та форматом .hdf5 (Рис. 4.16). Після чого у користувача з’являється можливість завантажити створений файл для можливості його аналізу у інших системах.

Рисунок 4.16 – Перетворення архіву даних до формату HDF5

Після того як усі частини набору даних будуть перетворені у формат hdf5 у контексті набору даних з’являється кнопка «Generate HDF5». При натисканні на неї створені файли .hdf5 об’єднуються до єдиного файлу набору даних у форматі .hd5 (Рис. 4.17) та, у користувача, з’являється можливість завантажити згенерований файл для використання у інших системах.

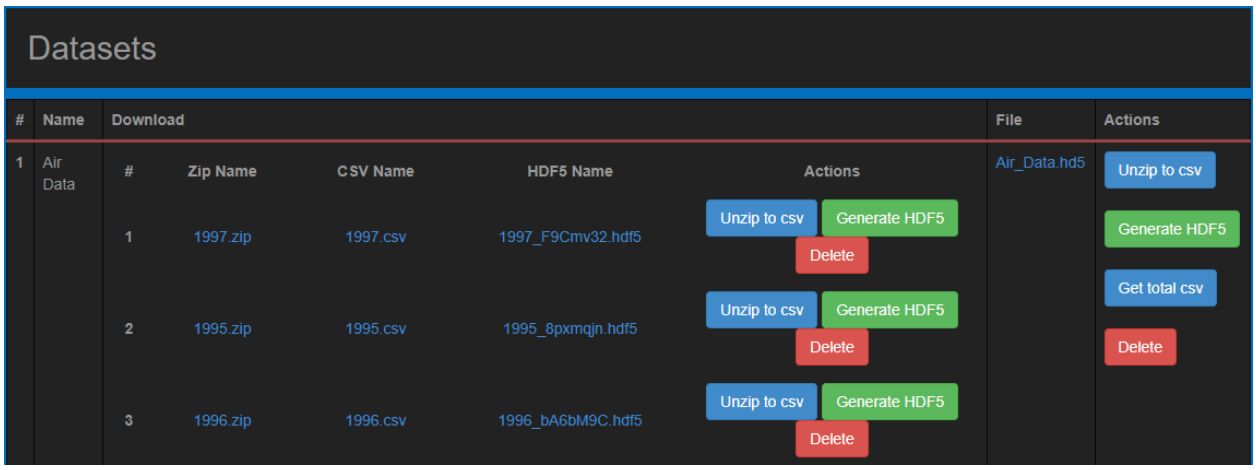


Рисунок 4.17 – Вихідний файл набору даних у форматі hd5

Для виконання запитів до набору даних було створено сторінку під назвою «Запити». На ній знаходиться список наборів даних користувача, для яких згенерований файл даних у форматі .hd5. Дані надаються у вигляді таблиці (Рис. 4.18).

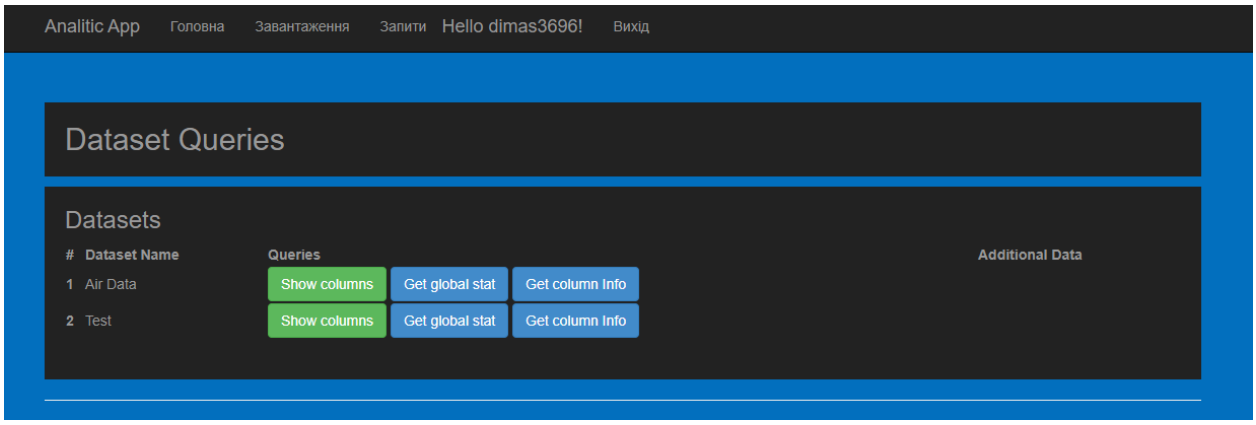


Рисунок 4.18 – Таблиця з наборами даних, готовими для аналізу

На даній сторінці передбачено три методи, які виконуються у контексті набору даних а якому вони знаходяться.

Відображення колонок

Даний метод виконує запит до набору даних та отримує перелік назв колонок у відповідь для подальшого здійснення запитів до них. Результуючий список колонок представлений у таблиці (Рис. 4.19).

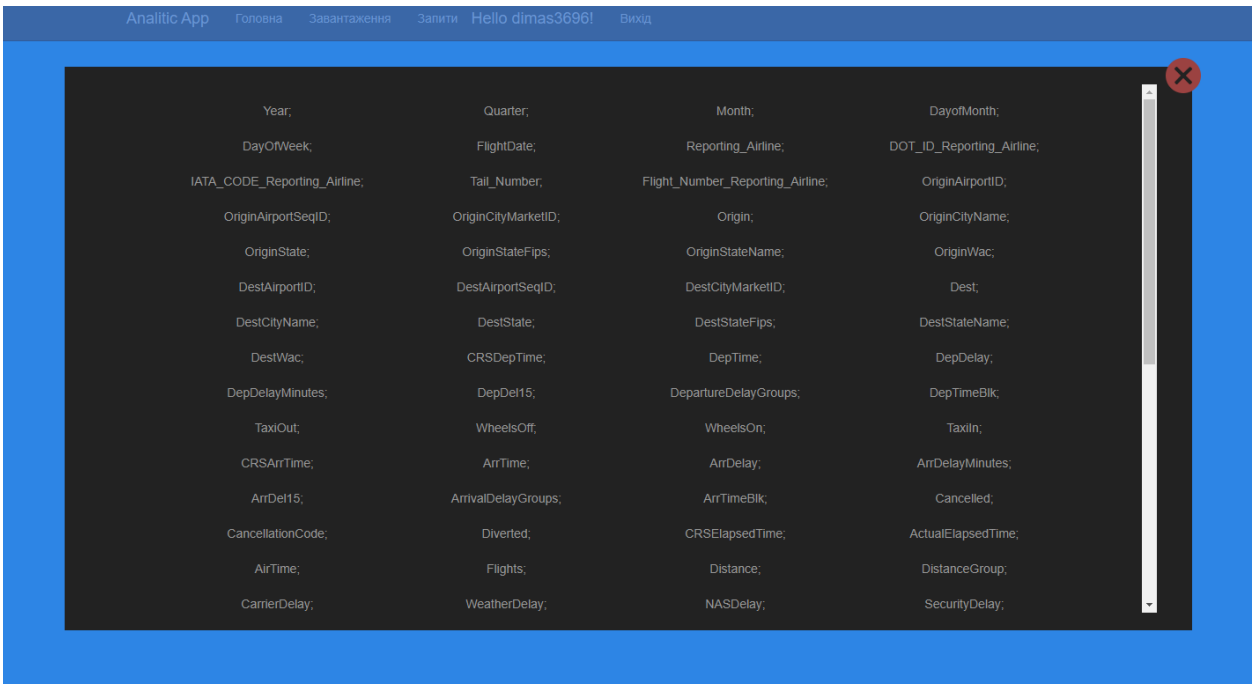
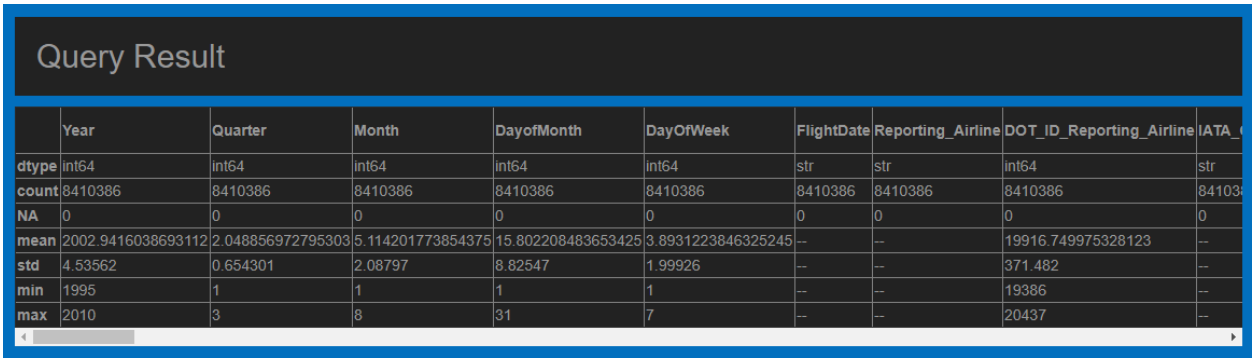


Рисунок 4.19 – Список колонок обраного набору даних

Отримання глобальної статистики

При виконанні цього запиту, до обраного набору даних, застосовується метод describe(). Після успішного виконання на стороні сервера, користувача перенаправляє на сторінку результатів запиту та відображається загальна інформація набору даних у табличному вигляді (Рис. 4.20). Таблиця генерується на стороні сервера.



	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_ID_Reporting_Airline	IATA_
dtype	int64	int64	int64	int64	int64	str	str	int64	str
count	8410386	8410386	8410386	8410386	8410386	8410386	8410386	8410386	84103
NA	0	0	0	0	0	0	0	0	0
mean	2002.9416038693112	2.048856972795303	5.114201773854375	15.802208483653425	3.8931223846325245	--	--	19916.749975328123	--
std	4.53562	0.654301	2.08797	8.82547	1.99926	--	--	371.482	--
min	1995	1	1	1	1	--	--	19386	--
max	2010	3	8	31	7	--	--	20437	--

Рисунок 4.20 – Глобальна статистика набору даних

Отримання загальної інформації за ім’ям колонки

При виконанні цього запиту, відображається поле для вводу імені колонки (Рис. 4.21) та формується AJAX запит до серверу, на якому формується таблиця з загальними даними по введеному імені колонки (Рис. 4.22). До її складу входять такі поля як мінімальне, максимальне та середнє значення числової колонки.

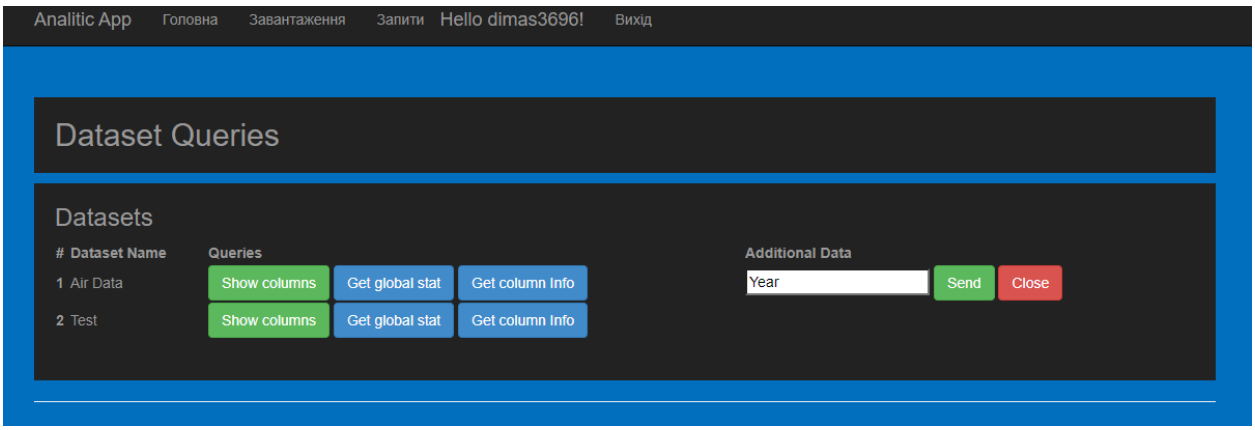


Рисунок 4.21 – Форма вводу імені колонки

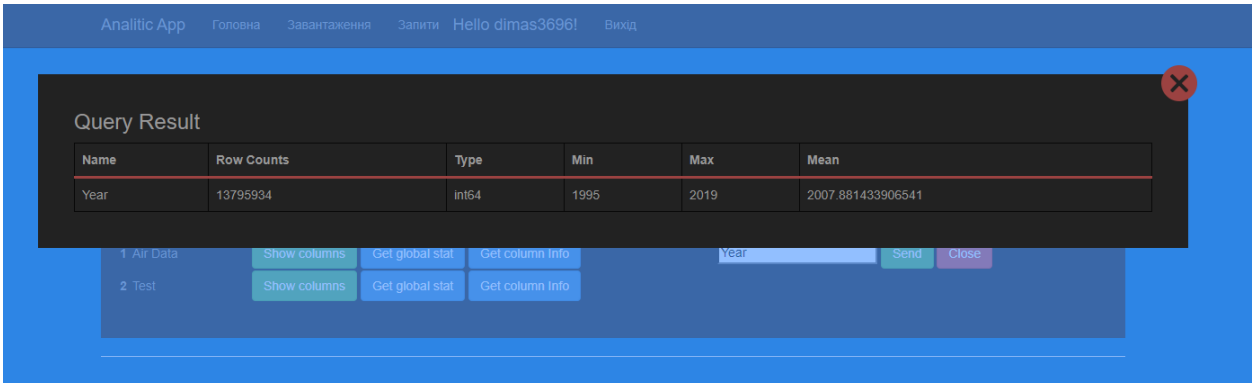


Рисунок 4.22 – Загальна статистика за обраною колонкою

4.2 Налаштування Apache Spark

Для аналізу даних за допомогою технології Apache Spark було обрано інструмент databricks, який входить до складу хмарних технологій Microsoft Azure.

Для початку роботи з інфраструктурою Apache Spark необхідно сформувати файл даних у форматі .csv та завантажити його на хмарну платформу. Розмір вхідного набору даних достатньо великий, тому було вирішено скористатися сховищем BLOB-об’єктів Azure. При використанні тимчасового облікового запису Azure існує обмеження на розмір файлу, який планується завантажити, у 40 гігабайтів.

Після завантаження файлу до контейнеру необхідно отримати ключ, для подальшої роботи з ним (Рис 4.23).

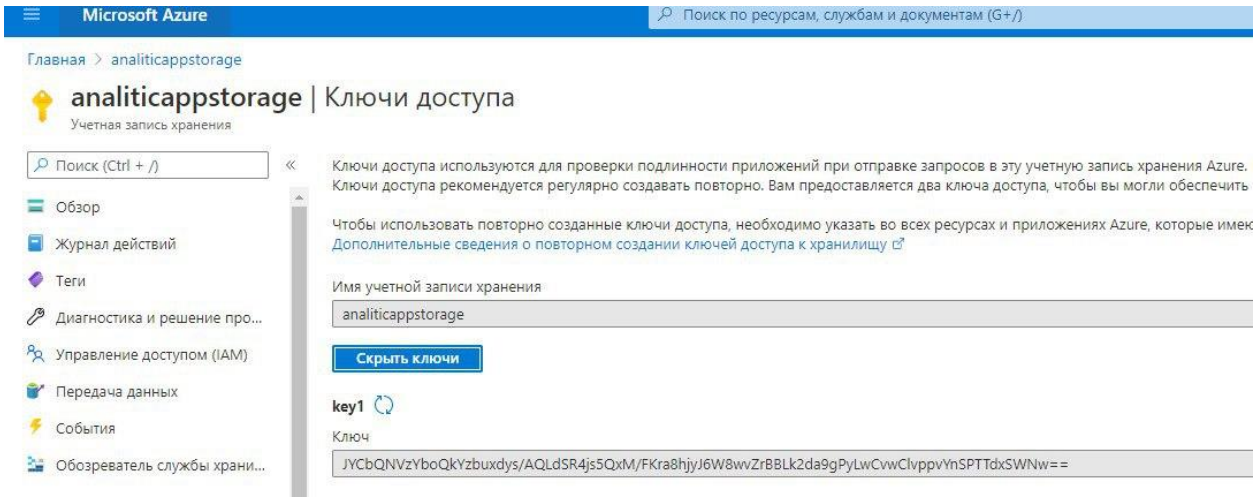


Рисунок 4.23 – Ключ доступу до контейнеру BLOB-об’єктів

Паралельно з цим, необхідно створити кластер у середовищі databricks (Рис. 4.24). У даному випадку був створений кластер від 2 до 8 працівників з виділенням 14 гігабайтів оперативної пам’яті, 4 ядер та 0,75 DBU на кожного.

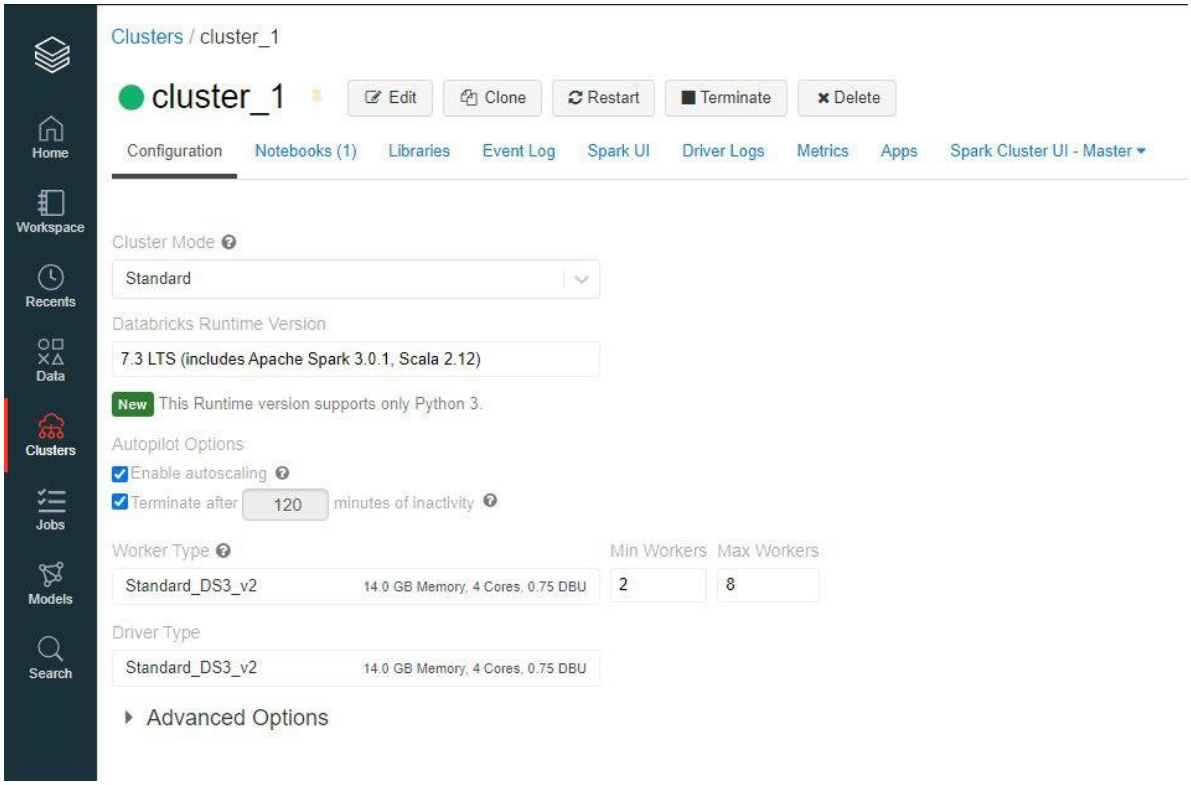


Рисунок 4.24 – Створений кластер

Після створення кластеру необхідно додати до нього блокнот у якому будуть виконуватися запити до набору даних. Мовою програмування блокноту було обрано Python.

Далі необхідно підключитися до контейнеру BLOB-об’єктів, використовуючи попередньо згенерований ключ (Рис. 4.25).

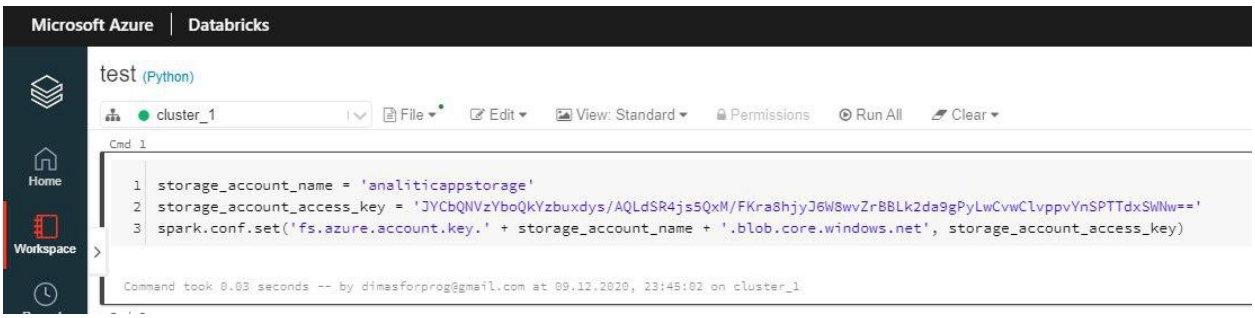


Рисунок 4.25 – Підключення до контейнеру Blob-об’єктів

Потім необхідно відкрити файл набору даних за допомогою Apache Spark та відобразити записи з набору для перевірки правильності підключення (Рис. 4.26).

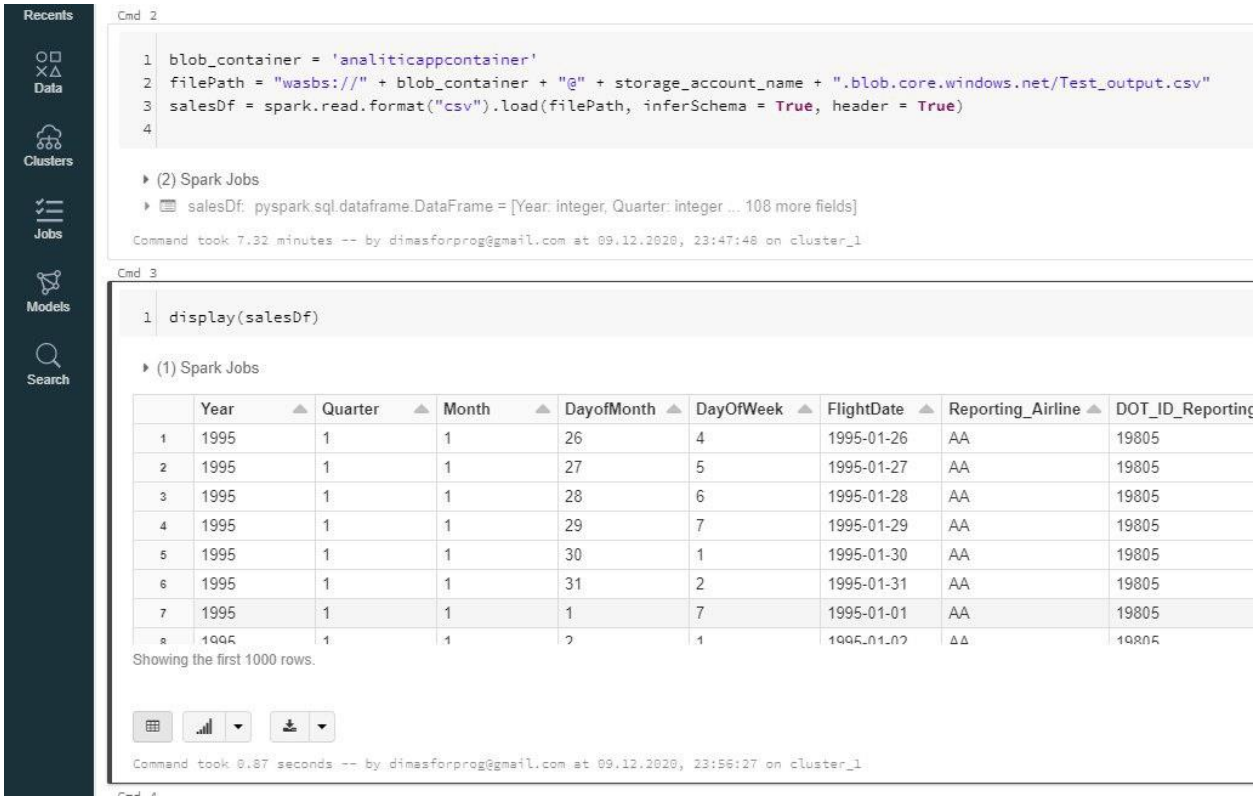


Рисунок 4.26 – Відкриття набору даних за допомогою Apache Spark

Після здійснення всіх вищеперерахованих операцій можна тестувати швидкодію виконання методів у середовищі Apache Spark (Рис. 4.27).

Рисунок 4.27 – Виконання запитів до набору даних у середовищі Spark

4.3 Порівняння швидкодії

Створивши веб додаток з запропонованою архітектурою та розгорнувши Apache Spark у хмарному середовищі та локально, було проведено ряд досліджень з метою порівняння швидкості виконання однакових запитів у цих середовищах.

Першим кроком була проведена оцінка швидкості завантаження даних. Результати наведені у таблиці 4.1.

Таблиця 4.1 – Швидкість завантаження даних

Етап	Розроблена архітектура	Apache Spark (локальний)	Apache Spark (у хмарному середовищі)
Завантаження даних на сервер	46 хвилин	-	4 години 46 хвилин

Продовження таблиці 4.1

Етап	Розроблена архітектура	Apache Spark (локальний)	Apache Spark (у хмарному середовищі)
Підготовка даних (приведення до необхідного формату)	2 години 17 хвилин	-	-
Підсумки	3 години 3 хвилини	-	4 години 46 хвилин

Під час використання розробленого додатку необхідно провести мінімум два етапи підготовки даних – завантажити частини наборів та розархівувати їх на стороні серверу, а потім об’єднати .csv файли до єдиного файлу даних у форматі .hd5. Якщо користуватися локально розгорнутою інфраструктурою Apache Spark необхідність у завантажені даних відпадає, так як дані вже знаходяться на локальному пристрої. Також, якщо розглянути завантаження файлів до Apache Spark, яка розгорнута у хмарному середовищі, необхідно провести лише один етап – це завантажити дані до сховища BLOB-об’єктів Azure, після чого вони стають доступними у середовищі DataBricks.

Підсумовуючи вищезазначене стає зрозумілим, що розроблена архітектура працює більш швидко у порівнянні з розгорнутому у хмарному середовищі Apache Spark, але потребує більшої кількості кроків від користувача.

Далі було проведено порівняння швидкості відкриття набору даних для подальшої роботи з ним. Результати тесту наведені у таблиці 4.2.

Таблиця 4.2 – Порівняння швидкості відкриття набору даних.

Розроблена архітектура	Apache Spark (локальний)	Apache Spark (у хмарному середовищі)
0:00:00.813978	0:52:02.785845	0:00:07:32

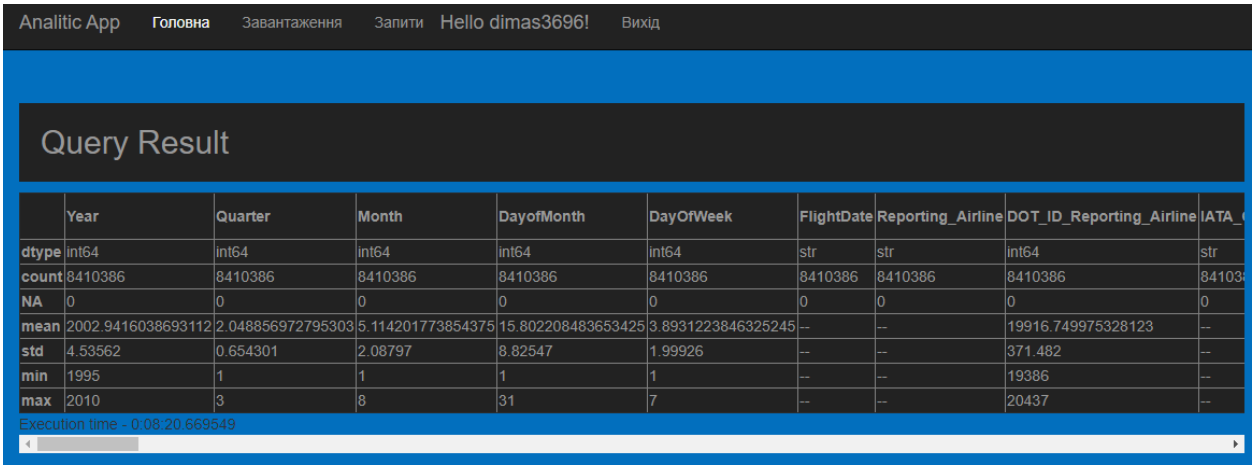
В результаті тесту, було підтверджено ефективність використання методу memory mapping, який використовується у розробленій архітектурі та просто екранує файл до оперативної пам’яті, завдяки чому файл набору даних

відкривається майже моментально. На відміну від цього, локально розгорнута інфраструктура Apache Spark використовує значно більшу кількість ресурсів та часу.

Також було проведено порівняння швидкості отримання глобальної статистики за набором даних. Його результати наведені у таблиці 4.3.

Таблиця 4.3 – Порівняння швидкості отримання глобальної статистики.

Розроблена архітектура	Apache Spark (локальний)	Apache Spark (у хмарному середовищі)
0:08:20.669549 (Рис. 4.28)	-	0:48:17.00 (Рис. 4.29)



	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_ID_Reporting_Airline	IATA_
dtype	int64	int64	int64	int64	int64	str	str	int64	str
count	8410386	8410386	8410386	8410386	8410386	8410386	8410386	8410386	84103
NA	0	0	0	0	0	0	0	0	0
mean	2002.9416038693112	2.048856972795303	5.114201773854375	15.802208483653425	3.8931223846325245	--	--	19916.749975328123	--
std	4.53562	0.654301	2.08797	8.82547	1.99926	--	--	371.482	--
min	1995	1	1	1	1	--	--	19386	--
max	2010	3	8	31	7	--	--	20437	--

Рисунок 4.28 – Виконання запиту за допомогою розробленої архітектури

Рисунок 4.29 – Виконання запиту у середовищі DataBricks

З проведеного порівняння стає очевидним, що швидкість виконання запитів за допомогою розробленої архітектури є досить високою при всій економії, яку вона надає.

4.4 Вимоги до технічного забезпечення

Для розгортання запропонованого рішення віддалений сервер має володіти мінімальними технічними можливостями, такими як:

– процесор з тактовою частотою не нижче 1,3 ГГц;
– кількість ядер процесора – 1;
– об’єм HDD або SSD не менший за 200Гб;
– об’єм RAM не менший за 1024Мб;

Для комфортного користування розробленим архітектурним рішенням бажано, щоб система відповідала наступним вимогам:

– процесор з тактовою частотою від 1,6 Ггц;
– кількість ядер процесора – 2;
– об’єм HDD або SSD 1Тб або більше;
– об’єм RAM не менший за 2048Мб

Висновки до розділу

У цьому розділі було продемонстровано як застосовується запропонована архітектура, детально описано процес створення веб додатку з її застосуванням. Було наведено схему бази даних та детально описані всі таблиці і стовпці в ній.

Також було проведено порівняльний аналіз швидкості виконання запитів з подібними, більш відомими технологіями та продемонстровано перевагу запропонованої архітектури з точки зору економічності та швидкості.

На останок, було розраховано мінімальні та комфортні технічні характеристики серверного комп’ютера, на якому буде відбуватися розгортання запропонованої архітектури.

5 РОЗРОБКА СТАРТАП ПРОЕКТУ

5.1 Опис ідеї проекту

Проаналізуємо зміст ідеї, її можливі напрямки застосування, чим запропонована ідея відрізняється від існуючих аналогів, а також основні вигоди, які може отримати користувач продукту. Результати аналізу представлені у таблиці 5.1.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигода для користувачів
Розробити веб додаток для загального аналізу набору даних	Завантаження файлів	Можливість часткового завантаження файлів у вигляді zip архівів
	Конвертування файлів	Приведення архівів у формати .csv, .hdf5 та .hd5
	Об’єднання файлів у єдиний набір	Швидке об’єднання файлів у єдиний вихідний файл у форматах .csv, .hdf5 та .hd5
	Керування наборами даних та їх частинами	Зручні методи додавання/видалення частин наборів даних з датасетів та наборів даних в цілому
	Виконання запитів	Швидке виконання агрегатних запитів до набору даних без використання оперативної пам’яті серверного комп’ютера

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Можливі конкуренти			W(слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Запропонована архітектура	Apache Spark	pandas			
1	Робота з архівами даних	+	-	-			+
2	Обробка частин наборів даних	+	-	-			+
3	Можливість масштабування	-	+	-	+		
4	Обробка файлів без використання RAM	+	-	-			+

Продовження таблиці 5.2

№	Техніко-економічні характеристики ідеї	Можливі конкуренти			W(слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Запропонована архітектура	Apache Spark	pandas			
5	3D візуалізація даних	-	-	+	+		
6	Конвертування форматів	+	-	+		+	
7	Веб інтерфейс управління	+	+	-		+	

Основними відмінностями є спосіб зберігання та обробки файлів наборів даних, а також методи завантаження даних на сервер.

5.2 Технологічний аудит ідеї проекту

В даному підрозділі наведені технології, за допомогою яких можна реалізувати проект. Результат наведено у таблиці 5.3.

Таблиця 5.3 – Технології для реалізації ідеї проекту

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Завантаження файлів у вигляді заархівованих частин набору даних	Серверний компонент. На вході приймає набір даних заархівованих у форматі .zip. Після завантаження на сервер перетворює їх у .csv файли з подальшою можливістю конвертації у формат для аналізу даних .hdf5	Необхідні: Python, ZipFile, Django	Доступні, безкоштовні
2	Додавання/видалення частин наборів даних			
3	Об’єднання частин наборів даних			
4	Конвертація частин даних у формат для аналізу .hdf5	Серверний компонент. На вході приймає ідентифікатор набору даних чи частини набору даних, на виході створюється конвертований або об’єднаний файл та додає його до БД.	Необхідні: Python, Vaex, WebRoot, Django	Доступні, безкоштовні
5	Об’єднання конвертованих файлів у загальний файл датасету для аналізу з форматом .hd5			
6	Об'єднання частин наборів даних у єдиний вихідний файл в форматі .csv			

Продовження таблиці 5.3

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
7	Виконання запитів до набору даних	Серверний компонент. На вхід приймає ідентифікатор набору даних та запит для аналізу	Необхідні: Python, Vaex, Django Templates, Pandas	
8	Отримання результатів виконання запитів			
9	Збереження наборів даних	Серверний компонент. Запис файлів даних до бази даних, видалення/додавання наборів даних та їх частин	Необхідні: Python, Django, SQLite DB	Доступні, безкоштовні
10	Управління наборами даних			
Обрані технології реалізації ідеї проекту: Python, Vaex, ZipFile, Pandas, Django, SQLite. Обрані технології є фактично набором інструментів та фреймворків для реалізації мети додатку, не потребують доробки.				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 5.4 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Економний аналіз великих об’ємів даних	Невеликі та середні компанії та приватні підприємці, які мають необхідність в аналізі даних, але не мають значних фінансів для здійснення цих операцій	- Наявність фінансового забезпечення - Різний об’єм необхідних для аналізу даних	- Стабільність - Постійна підтримка - Впровадження оновлень
Аналіз даних на пристроях з низькими технічними можливостями	Люди, які зацікавлені в аналізі даних, але не мають можливість оплати хмарних технологій		

Продовження таблиці 5.4

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Зручне керування завантаженими наборами даних	Середні за розміром та невеликі компанії, а також фізичні особи - підприємці	- Наявність фінансового забезпечення Різний об’єм необхідних для аналізу даних	- Стабільність - Постійна підтримка Впровадження оновлень

Також необхідно оцінити фактори, які можуть завадити реалізації проекту. Оцінку наведено у таблиці 5.5.

Таблиця 5.5 – Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Поява конкурентів	Можлива поява конкурентів, які спроможуться створити більш якісний продукт або розширити вже популярний та існуючий сервіс. Можлива поява більш продуктивних продуктів	Удосконалення існуючого рішення новими технологіями, розробка альтернативних методів аналізу даних з використанням технології екранування пам’яті

Продовження таблиці 5.5

Фактор	Зміст загрози	Можлива реакція компанії
Значне підвищення вартості оренди дискового простору	Можливе підвищення вартості HDD пам’яті через неактуальність використання та зменшення конкуренції на надавання послуг хостингами	Перехід на більш досконалий та актуальний на той час метод зберігання даних без використання оперативної пам’яті серверного комп’ютера
Збільшення переваги хмарних сервісів	Здешевлення та пришвидшення обчислень з використанням хмарних технологій, поява нових технологій для обчислень надвеликих об’ємів статистичних запитів з новими потужними технологіями	Пошук нових методів та бібліотек для аналізу надвеликого масиву даних
Здешевлення оперативної пам’яті	Хмарні обчислення стануть дешевшими та будуть більш гнучкими, ніж запропоноване архітектурне рішення, а також звичайні комп’ютери користувачів будуть обладнані додатковим об’ємом ОЗУ	Впровадження більш зручних методів маніпулювання наборами даних та перехід на гібридний варіант аналізу даних з оптимальним використанням дискового простору серверного комп’ютера та його ОЗУ

Продовження таблиці 5.5

Фактор	Зміст загрози	Можлива реакція компанії
Відсутність репутації сервісу	Середні та малі за обсягом компанії можуть боятися завантажувати на невідомий сервіс свої статистичні дані, так як бояться, що ці дані зможуть потрапити до рук конкурентів	Розробка функціоналу шифрування даних та отримання сертифікату безпеки від стороннього та відомого аудитора

У таблиці 5.6 представлено фактори можливості системи.

Таблиця 5.6 – Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Невелика кількість конкурентів	На сьогоднішній день на ринку обробки надвеликих об’ємів даних існує невелика кількість конкурентів, їх програмні продукти в переважній більшості використовують горизонтальне збільшення кількості кластерів для збільшення обчислювальної потужності системи	Розвиток продукту у напрямку мінімізації використання RAM та зниження вартості обчислень

Продовження таблиці 5.6

Фактор	Зміст можливості	Можлива реакція компанії
Можливість побудови власної репутації	Новий сервіс на ринку обчислення великих об’ємів даних має всі можливості побудови власної репутації без попередньої історії	Пошук замовників, розповсюдження інформації про створену платформу, мінімальна націнка на вартості обчислень
Демонстрація вигоди використання розробленого додатка	Демонстрація потужності та економічності розробленого сервісу у порівнянні з конкурентами	Створення веб сторінки додатку з інформацією про алгоритми роботи та ефективність обраних методів, а також візуалізація графіків, порівняння швидкодії виконання запитів до аналогічних наборів даних у розробленому середовищі та середовищі конкурентів
Демонстрація простоти користування розробленим додатком	Швидке занурення до інтерфейсу аналізу наборів даних	Розробка безкоштовних тимчасових підписок для звертання уваги потенційних користувачів на можливість використання створеної системи
Зростання попиту на системи аналізу даних	Збільшення клієнтської бази	Привертання уваги потенційних користувачів та реклама

Основними загрозами у майбутньому є здешевлення хмарних обчислень та створення інтерфейсів, зрозумілих для звичайного користувача, що можуть негативно вплинути на кількість потенційних клієнтів. Для запобігання даної загрози необхідно з кожним релізом збільшувати швидкість виконання запитів та роботи системи у цілому, використовуючи нові та ефективні методи.

Важливим фактором виходу на ринок є аналіз ринку пропозиції. Для цього потрібно визначити загальні риси конкуренції на ринку. У таблиці 5.7 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 5.7 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - чиста	Можливість вільно конкурувати на ринку	Можливість чесної конкуренції
2. За рівнем конкурентної боротьби – міжнародний	На ринку присутні іноземні фірми-конкуренти	Розробити додаток англійською мовою для збільшення потенційних користувачів на міжнародному ринку
3. За галузевою ознакою – міжгалузева	Розроблена архітектура може бути використана в різних галузях	Підтримка уніфікованих агрегатних запитів до різноманітних наборів даних
4. Конкуренція за видами товарів – товарно-видова	Види товарів схожі за функціональністю	Враховувати недоліки методів обробки даних
6. За характером конкурентних переваг – цінова	Ціна - дуже важливий фактор при виборі продукту	Приділяти більше уваги вартості розробки та впровадження
7. За інтенсивністю – немарочна	Бренд не впливає на впізнаваність продукції	

Також, для кращого розуміння ринку, необхідно провести детальний аналіз умов конкуренції в галузі. Результати проведеного аналізу наведені у таблиці 5.8.

Таблиця 5.8 – Аналіз конкуренції в галузі за М.Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Azure Databriks	Google Cloud	Значна кількість постачальників	Активно диктують умови	Збільшення кількості аналогічних товарів
Висновки:	Доволі відома компанія, тому конкуренція інтенсивна	Компанія має потенціал запуску технологій аналізу великих масивів даних	Постачальник не впливає на ринок	Мають основний вплив	Велика кількість товарів замінників

За результатами аналізу конкурентів зроблено висновок про можливість роботи на ринку, навіть при наявній конкуренції. Для подальшого розвитку проекту необхідно покращити ефективність роботи методів обробки даних. Дані результати були враховані при порівняльному аналізі факторів конкурентоспроможності в таблиці 5.9.

Таблиця 5.9 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Розроблене архітектурне рішення спрямоване на зниження ціни обробки великих об’ємів даних шляхом використання методів екранування пам’яті.

Продовження таблиці 5.9

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
2	Швидкість виконання запитів	Використання методів віртуальних стовпців для аналізу даних та виконання запитів «на льоту», що добре впливає на швидкість виконання запитів, не копіюючи та дублюючи при цьому дані у пам’яті серверного комп’ютера
3	Швидкість завантаження наборів даних	Можливість обробки заархівованих наборів даних та їх частин
4	Гнучкість маніпулювання набором даних	Можливість додавати та видаляти частини набору даних та формувати з них результуючі файли для подальшого аналізу

Використовуючи дані конкурентоспроможності проекту, необхідно проаналізувати сильні і слабкі сторони проекту (таблиця 5.10) та навести SWOT-аналіз стартап-проекту (таблиця 5.11).

Таблиця 5.10 – Порівняльний аналіз сильних та слабких сторін архітектурного рішення для аналізу великого обсягу статистичних даних на пристроях з низькими технічними можливостями

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з хмарними технологіями для даних						
			-3	-2	-1	0	+1	+2	+3
1.	Ціна	20		+					
2.	Швидкість виконання запитів	14					+		

Продовження таблиці 5.10

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з хмарними технологіями для даних						
			-3	-2	-1	0	+1	+2	+3
3.	Швидкість завантаження наборів даних	16					+		
4.	Гнучкість маніпулювання набором даних	12			+				

Таблиця 5.11 – SWOT-аналіз стартап-проекту

Сильні сторони: Комбінація рішень для швидкого завантаження даних на сервер, невелике використання дорогих компонентів серверу, що призводить до дешевизни продукту, оптимальні методи обробки статистичних запитів	Слабкі сторони: Вихід на ринок невідомою компанією, що потребує потужної рекламної компанії
Можливості: Покращення методів аналізу даних, розробка нового функціоналу керування набором даних	Загрози: Здешевлення вартості надання послуг хмарними сервісами. Поява конкурентів, які будуть використовувати такі ж методи обробки даних. Можливе подорожання вартості жорстких дисків.

Сильними сторонами розробленого проекту є низька вартість обчислень, висока швидкість виконання операцій та гнучкість роботи з набором даних. Слабкими сторонами є недовіра до подібних проектів на перших етапах та наявність більш відомих конкурентів.

На основі SWOT-аналізу було розроблено альтернативи ринкової

поведінки для виведення стартап-проекту на ринок. Далі необхідно проаналізувати строки реалізації та ймовірність отримання ресурсів. Результати аналізу наведені у таблиці 5.12.

Таблиця 5.12 – Альтернативи ринкового впровадження стартап-проекту.

№ п/п	Альтернатива (орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання результатів	Строки реалізації
1	Створення мінімального продукту з набором найголовніших функцій та швидкий вихід на ринок	Висока	Від 2 до 5 місяців
2	Створення потужного сервісу з великою кількістю різноманітних операцій з даними	Середня	Від 8 до 12 місяців

Було розглянуто дві основні стратегії – створення мінімального продукту для виходу на ринок для оцінки відгуків від користувачів, та створення продукту, що буде містити весь запланований функціонал. Обраною альтернативою є перша, тому що дуже важливим фактором вірного напряму розвитку проекту є відгуки та побажання користувачів. Реалізувавши основний функціонал системи у першу чергу, ми зможемо зрозуміти яких додаткових функцій не вистачає майбутнім користувачам та зможемо задовільнити їх потреби, не витрачаючи час на реалізацію методів, якими вони не будуть користуватися.

5.4 Розроблення ринкової стратегії проекту

Для створення ринкової стратегії проаналізуємо цільові групи потенційних користувачів. Результати аналізу наведені у таблиці 5.13.

Таблиця 5.13 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Фізичні особи підприємці	Середня	Середній	Низька	Низька
2	Компанії малого бізнесу	Висока	Високий	Низька	Низька
3	Компанії середнього бізнесу	Середня	Низький	Висока	Середня
Які цільові групи обрано: в першу чергу необхідно звернути увагу на компанії малого бізнесу, оскільки вони не мають можливості витрачати великі суми коштів на аналіз даних, але відчувають гостру необхідність.					

На основі отриманої інформації розробимо базову стратегію розвитку. Результати аналізу наведено у таблиці 5.14.

Таблиця 5.14 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Вихід на профільний ринок з сервісом аналізу великих об’ємів даних	Масовий маркетинг серед компаній малого бізнесу	Комбінація рішення для аналізу даних при низькому використанні дорогої ОЗУ та зручні механізми маніпулювання даними, а також необмежений об’єм вхідних наборів даних	Стратегія лідерства по витратах

На наступному етапі необхідно обрати стратегії конкурентної поведінки. На основі базової стратегії розвитку розробимо стратегію конкурентної поведінки (таблиця 5.15).

Таблиця 5.15 – Визначення базової стратегії конкурентної перевірки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні, проект не є першопрохідцем на ринку, але володіє унікальними методами обробки наборів даних	Компанія буде пропонувати свій продукт, як новим споживачам у даній сфері, так і тим, хто вже користується сервісами конкурентами	Компанія не буде копіювати конкурентів	Стратегія наслідування лідеру

За стратегію конкурентної поведінки було обрано стратегію наслідування лідеру, за допомогою якої вдасться уникнути боротьби з лідерами ринку.

За допомогою проведених досліджень визначена стратегія позиціонування з урахуванням основних вимог до проекту та стратегій його розвитку, які наведені у таблиці 5.16.

Таблиця 5.16 – Визначення стратегій позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Низька ціна, швидкість, функціональні можливості	Стратегія лідерства по витратах	Низька вартість обробки даних, висока швидкість, зручний інтерфейс роботи з датасетами	Економічність, простота, швидкість, зручність

Отже, основою ринкової стратегії буде залучення уваги малих бізнес компаній, що відчувають необхідність у аналізі даних для подальшого розвитку на ринку, але не мають достатнього фінансового забезпечення для його проведення.

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.17 підсумовуються результати попереднього аналізу конкурентоспроможності проекту.

Таблиця 5.17 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Дешевий аналіз статистичних даних	Мінімізація використання ОЗУ серверного пристрою, що призводить до здешевлення отримання результатів аналізу набору даних	Всі конкуренти спрямовані на збільшення використання ОЗУ із збільшенням об’єму даних, який підлягає аналізу, що призводить до значного збільшення вартості обчислень
2	Швидкі методи завантаження датасетів на сервер	Можливість завантажувати архіви і частини даних на сервер, що призводить до швидшої та більш гнучкої моделі маніпулювання даними	Зазвичай використовують швидкі методи завантаження даних на сервер, але не розбивають їх на частини, через що, при завантаженні дійсно великого файлу даних доволі часто відбувається збій. Це викликано не стабільним інтернет з’єднанням користувача
3	Керування наборами даних	Зручність обробки і управління наборами даних, що досягається шляхом розробки інтуїтивно зрозумілого інтерфейсу користувача	Схожі системи розраховують на те, що користувач добре володіє знаннями про комп’ютер та хоча б трохи розуміється на їх системі, що призводить до значних витрат часу на ознайомлення з інфраструктурою проекту

Надалі розробляється тривіальна маркетингова модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 5.18).

Таблиця 5.18 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Віддалений доступ та управління набором даних, низька вартість отримання результатів запитів, висока швидкість аналізу даних
II. Товар у реальному виконанні	Властивості/характеристики
	1. Сервер, що розташований на хостингу, з великою кількістю жорсткого дискового простору та малим використанням ОЗУ. 2. Веб-клієнт для доступу до серверу, завантаження та маніпулювання даними, а також виконанням запитів до набору даних.
	Марка: назва організації-розробника + назва товару
III. Товар із підкріпленням	До продажу: базова версія
	Після продажу: постійна модернізація
За рахунок чого потенційний товар буде захищено від копіювання: методи обробки та аналізу наборів даних, а також методи їх зберігання на сервері. Тобто захист ідеї товару буде відбуватися шляхом захисту інтелектуальної власності на комплексне поєднання властивостей і характеристик.	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановлені ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари замінники, а також аналіз рівня доходів цільової групи споживачів (таблиця 5.19). Аналіз проводиться експертним методом.

Таблиця 5.19 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Ціна починається від 0,4$ - 0.55$ за годину використання та залежить від технологій, які входять у конкретний пакет. При оформленні пакету на рік, можливі знижки у розмірі від 6% до 33%. Мінімальний тарифний план на рік дорівнює 25 тис. дол., а максимальний – 2 млн. дол. та залежить від кількості виконаних запитів та задіяного дискового простору.	Хмарні платформи в залежності від об’єму набору даних, потужності пристроїв обчислення та кількості операцій починаються від 504 дол. на місяць і можуть досягати 14.5 тис. дол. на місяць.	1 тис. дол – 7 тис. дол	200 дол. – 700 дол.

Наступним кроком є визначення оптимальної системи збуту (таблиця 5.20).

Таблиця 5.20 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти прагнуть отримувати товар за меншою ціною за рахунок конкуренції	Постійний доступ до сервісу	Однорівневий чи нульовий канал збуту	Власноруч та через посередників

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування та визначену специфіку поведінки клієнтів (таблиця 5.21).

Таблиця 5.21 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Аналіз даних у Excel	Інтернет, телефон	Моніторинг та прогнозування	Звернути увагу на ціну, зручність та швидкість порівняно з конкурентами	Дешевий та якісний аналіз даних

Висновки до розділу

Основною ідеєю проекту є створення сервісу для аналізу великого об’єму статистичних даних із зручним інтерфейсом маніпулювання наборами даних та економічним швидким виконанням запитів у розробленому середовищі.

Основні технології, що будуть використані для реалізації проекту є технології та бібліотеки: Python, Vaex, ZipFile, Pandas, Django, SQLite. Обрані технології є фактично набором інструментів та фреймворків для реалізації мети додатку, не потребують доробки.

Аналіз майбутнього ринку збуту показав, що на сьогодні попит на аналіз даних зростає з неймовірною швидкістю. Розуміння своїх клієнтів та їх потреб є важливим фактором для будь якої компанії, тому ця галузь користується попитом майже у всіх сферах бізнесу.

Головним фактором конкурентоспроможності є невисока ціна за використання сервісу та надання прийнятної швидкості виконання запитів для задоволення потреб потенційних користувачів.

Потенційною цільовою аудиторією користувачів розробленого проекту є малий бізнес, який не спроможний сплачувати за потужні сервіси хмарних обчислень та має необхідність у аналізі даних для подальшого розвитку на ринку.

Основною стратегією є стратегія лідерства за ціновим сегментом на ринку, тобто надання потенційним користувачам якісних послуг за невеликою вартістю.

Монетизація проекту буде відбуватися шляхом оформлення підписок, різниця між якими буде лише у обмеженні використання дискового простору аккаунтом користувачем.

У результаті аналізу, можна зробити висновок, що даний проект є досить перспективним, так як сфера аналізу даних зростає з кожним роком, а вартість обчислень лише збільшується. Існуючі альтернативні рішення є значно дорожчими у використанні і тому стають не досить привабливими для ведення бізнесу. Завдяки здешевленню аналізу даних можна залучити нові компанії, які раніше не мали можливості користування даною сферою.

ВИСНОВКИ

У ході виконання даної роботи було проведено дослідження сучасних методів аналізу великих об’ємів статистичних даних. Аналіз цих методів показав, що існують методи та технології, які успішно використовуються у сфері аналізу даних, але вони мають багато недоліків, зокрема для розглянутих бібліотек локального аналізу даних існує обмеження у об’ємі набору даних, який надається для обробки. Зі збільшенням об’єму цих даних значно збільшується час на отримання результатів, а у разі, якщо розмір файлу перевищує об’єм оперативної пам’яті комп’ютера, то аналіз такого набору даних стає зовсім неможливим. Якщо розглянути технології хмарних обчислень, то проблем з розміром вхідного файлу не виникає, але зі збільшенням об’єму цього файлу збільшується і кількість виділених ресурсів для його обробки, що значно впливає на вартість цих обчислень.

Зважаючи на вищезазначене, було сформовано цілі та мету задачі даної роботи та наведено опис інформаційного забезпечення. Також була детально описана архітектура програмного забезпечення та представлені необхідні технології для створення подібних систем.

Було проведено емпіричне дослідження з відомими та потужними на сьогоднішній день технологіями хмарних обчислень, зокрема Azure DataBricks Apache Spark та представлено результати заміру часу виконання аналогічних запитів у цих середовищах.

На останок, було проведено дослідження потенціалу продукту з комерційної точки зору, було визначено потенційних клієнтів, конкурентоспроможність продукту та його основні переваги та недоліки у порівнянні з існуючими рішеннями. За результатами цього дослідження було обрано стратегію поведінки на ринку, стратегію альтернативної поведінки на ринку та визначені методи монетизації розробленого продукту. Також були наведені основні напрямки подальшого розвитку ідеї для привертання уваги потенційних клієнтів та порівняно економічну доцільність із схожими

рішеннями.

За матеріалами дисертації було опубліковано одні тези доповіді на науковій конференції [29].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) David R., John G., John R. The Digitization of the World From Edge to Core. An IDC White Paper – #US44413318, 2018. 3 с.

2) Айвазян С.А. Программное обеспечение по статистическому анализу данных: методология сравнительного анализа и выборочный обзор рынка/ С.А.Айвазян, В.С.Степанов [Електронний ресурс]. – Режим доступу: http://pubhealth.spb.ru/SAS/STatProg.htm. 2017.

3) Василенко Ж.В. Программное обеспечение по статистическому анализу данных. Методология сравнительного анализа [Електронний ресурс]. – Режим доступу: http://www.giac.unibel.by/sm_full.aspx?guid =8313. 2017.

4) Вуколов Э.А. Основы статистического анализа. Практикум по статистическим методам и исследованию операций с использованием пакетов STATISTICA и EXCEL: учеб.пособ.- 2-е изд., испр. и доп.- М.: Форум, 2008.- 464 с.

5) Герасевич В.А., Современное программное обеспечение для статистической обработки биомедицинских исследований/ В.А. Герасевич, А.Р. Аветисов [Електронний ресурс]. – Режим доступу: http://miklebig.narod.ru/docum/statprog.htm. 2017.

6) Ермантраут Е. Р. Статистичний аналіз агрономічних дослідних даних в пакеті Statistica-6 / Е.Р. Ермантраут, О.І. Присяжнюк, І.Л. Шевченко.– К. : Поліграф Консалтинг, 2007. – 55 с.

7) Караєва Н. В. Еколого-економічна оптимізація виробництва: інформаційна підтримка прийняття рішень: конспект лекцій. – К.: НТУУ «КПІ», 2016. – 115 с.

8) Красильников Д.Е. Программное обеспечение эконометрического исследования [Електронний ресурс]. – Режим доступу: http://www.unn.ru/pages/issues/vestnik/99999999_West_2011_3(2)/37.pdf . 2017.

9) Левченко Л.О., Кілянчук О.П., Повханич О.Ю. Огляд програмних продуктів фінансово-економічного аналізу діяльності енергопідприємств [Електронний ресурс]. – Режим доступу: http://urss.knuba.edu.ua/files/zbirnyk-8/121-127.pdf. 2017.

10) Майборода Р.Є., Сугакова О.В. Статистичний аналіз даних за допомогою пакету STATISTICA [Електронний ресурс]. – Режим доступу: http://matphys.rpd.univ.kiev.ua/downloads/courses/mmatstat/StatAn.doc. 2017.

11) Chris S. "The history of Unix's confusing set of low-level ways to allocate memory". [Електронний ресурс]. – Режим доступу: https://utcc.utoronto.ca/~cks/space/blog/unix/SbrkVersusMmap. 2018.

12) Коржов В. Многоуровневые системы клиент-сервер. [Електронний ресурс]. - Режим доступу: https://www.osp.ru/nets/1997/06/142618. 1997.

13) Anderson J. An Intro to Threading in Python. [Електронний ресурс]. - Режим доступу: https://realpython.com/intro-to-python-threading/. 2019.

14) Chappell D. A Short Introduction to Cloud Platforms. [Електронний ресурс]. - Режим доступу: http://www.davidchappell.com/CloudPlatforms--Chappell.pdf. 2008. 7 с.

15) Маккинни У. Python и анализ данных. — ДМК Пресс, 2015. — 482 с.

16. Вандер Плас Дж. Python для сложных задач. Наука о данных и машинное обучение. — Питер, 2017. — 576 с.

17) Devlin J. Tutorial: Using Pandas with Large Data Sets in Python. [Електронний ресурс]. - Режим доступу: https://www.dataquest.io/blog/pandas-big-data/. 2017.

18) Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. — Вильямс, 2017. — 480 с.

19) Нуньес-Иглесиас Х., Уолт Ш., Дэшноу Х. Элегантный SciPy. — ДМК Пресс, 2018. — 266 с.

20) С. Риза, У. Лезерсон, Ш. Оуэн, Д. Уиллс. Spark для профессионалов: современные паттерны обработки больших данных. — Питер, 2017. — 272 с.

21) Уайт, Том. Hadoop. Подробное руководство. — 2-е. — СПб.: Питер, 2013. — 672 с.

22) Moldovan D., Copil G., Truong H., Dustdar S. On estimating actuation delays in elastic computing systems. Proceedings of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). 2013. с. 33–42.

23) Oswald K. Работа с данными по-новому: Pandas вместо SQL. 2019.

24) Rocklin M. "Dask: Parallel Computation with Blocked algorithms and Task Scheduling". Proceedings of the 14th Python in Science Conference. 2015. 126–132 с.

25) Pathak P. RIP Pandas: Time to introduce the Vaex. 2019.

26) JIT Compilation Techniques. Aycock. 2003. 98 с.

27) У. Чан, П. Биссекс, Д. Форсье. Django. Разработка веб-приложений на Python. 2009. — 456 с.

28) Grinberg M. Flask Web Development. Developing web applications with Python — O'Reilly Media, 2014—258 p.

29. Вальчук Д.В. Архітектурне рішення для обробки великих обсягів статистичних даних на пристроях з низькими технічними можливостями / Д.В. Вальчук, М.М. Головченко // V Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2020): Матеріали наукової конференції студентів, магістрантів та аспірантів, м. Київ, 27 листопада 2020.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

views.py

```
from django.shortcuts import render, redirect
from django.conf import settings
from django.core.files import File
from datetime import datetime
from django.http import HttpRequest
from django.http import HttpResponse
from .models import DataFile, DataSet, DataSetHdf5File
from .forms import FileUploadForm
from datetime import datetime, date, time
from django.http import JsonResponse

import zipfile
import numpy as np
import os, glob
import pandas as pd
import vaex
from tqdm import  tqdm_notebook as tqdm

def index(request):
    assert isinstance(request, HttpRequest)
    datasets = list(DataSet.objects.filter(user_id=request.user.id))

    for dataset in datasets:
        dataset_files = DataFile.objects.filter(dataset=dataset)
        dataset.files = dataset_files

        dataset.files_count = len(dataset_files)

        dataset.hdf5_count = 0

        for file in dataset_files:
            if file.hdf5_file:
                dataset.hdf5_count = dataset.hdf5_count + 1

        try:
            dataset_hdf5_file = DataSetHdf5File.objects.filter(dataset=dataset).last()
            if dataset_hdf5_file:
                dataset.hdf5 = dataset_hdf5_file.hdf5_file
        except DataSetHdf5File.DoesNotExist:
            dataset.hdf5 = None

    return render(
        request,
        'app/index.html',
        {
            'title':'Datasets',
            'data' : datasets,
            'year':datetime.now().year,
        }
    )

def upload_files(request):
    current_user_id = request.user.id

    user_datasets = DataSet.objects.filter(user_id=current_user_id)

    dataset_name = ''

    if user_datasets.count() > 0:
        last_dataset = user_datasets.reverse()[0]
```

```
        dataset_name = last_dataset.dataset_name
    else:
        dataset_name = ''

    data = { 'dataset_name' : dataset_name }

    if request.method == "POST":

        form = FileUploadForm(request.POST, request.FILES)

        files = request.FILES.getlist('file')
        dataset_name = request.POST['dataset_name']

        db_dataset = DataSet.objects.filter(user_id=current_user_id, dataset_name=dataset_name)

        dataset = {}

        if db_dataset.count() < 1:
            dataset = DataSet.objects.create(user_id=current_user_id, dataset_name=dataset_name, creation_date=datetime.now())
        else:
            dataset = db_dataset.reverse()[0]

        if form.is_valid():
            for f in files:
                file_instance = DataFile(dataset=dataset, file=f)
                file_instance.save()
        datasets = list(DataSet.objects.filter(user_id=request.user.id))

        for dataset in datasets:
            dataset_files = DataFile.objects.filter(dataset=dataset)
            dataset.files = dataset_files

        return redirect('home');
    else:
        form = FileUploadForm(initial=data)
    return render(request,
        'app/upload.html',
        {
            'form' : form,
            'title':'Завантаження даних',
            'year':datetime.now().year,
        })

def remove_dataset(request,dataset_id=None):
    object = DataSet.objects.get(id=dataset_id)
    object.delete()

    datasets = list(DataSet.objects.filter(user_id=request.user.id))

    for dataset in datasets:
        dataset_files = DataFile.objects.filter(dataset=dataset)
        dataset.files = dataset_files

    return redirect('home')

def remove_datafile(request, datafile_id=None):
    object = DataFile.objects.get(id=datafile_id)
    object.delete()

    datasets = list(DataSet.objects.filter(user_id=request.user.id))
```

```
    for dataset in datasets:
        dataset_files = DataFile.objects.filter(dataset=dataset)
        dataset.files = dataset_files

    return redirect('home')

def fixBadZipfile(zipFile):
    f = open(zipFile, 'r+b')
    data = f.read()
    pos = data.find('\x50\x4b\x05\x06') # End of central directory signature
    if (pos > 0):
        f.seek(pos + 22)  # size of 'ZIP end of central directory record'
        f.truncate()
        f.close()

def unzip_dataset(request, dataset_id=None):
    dataset = DataSet.objects.get(id=dataset_id)
    datafiles_zip = DataFile.objects.filter(dataset=dataset)

    CHUNK_SIZE = 500000

    for datafile in datafiles_zip:
        if datafile.csv_file != "":
            continue

        with zipfile.ZipFile(datafile.file, 'r') as parent_file:

            file_name = datafile.file.name[:-4]
            dataset_folder = dataset.dataset_name + "_csvs" + "/" + file_name
            parent_file.extractall("datasets/" + dataset_folder)
            root_folder = os.path.join(settings.MEDIA_ROOT, dataset_folder)
            all_files = glob.glob(os.path.join(root_folder, "*.csv"))

            output_file = os.path.join(root_folder, "output.csv")

            first_one = True
            for csv_file_name in all_files:

                if not first_one: # if it is not the first csv file then skip the header row (row 0) of that file
                    skip_row = [0]
                else:
                    skip_row = []

                chunk_container = pd.read_csv(csv_file_name, sep=',', chunksize=CHUNK_SIZE, skiprows = skip_row)
                for chunk in chunk_container:
                    chunk.to_csv(output_file, mode="a", index=False)
                first_one = False

            for csv_file_name in all_files:
                os.remove(csv_file_name)

            opened_file = open(output_file, "r")
            file_instanse = File(opened_file)
            datafile.csv_file.save(file_name + ".csv", file_instanse, save=True)

            opened_file.close()

            os.remove(output_file)
```

```
    return redirect('home')

def unzip_datafile(request, datafile_id):
    CHUNK_SIZE = 500000

    datafile = DataFile.objects.get(id=datafile_id)

    if datafile.csv_file != "":
        return redirect('home')

    dataset = datafile.dataset

    with zipfile.ZipFile(datafile.file, 'r') as parent_file:

        file_name = datafile.file.name[:-4]
        dataset_folder = os.path.join(dataset.dataset_name + "_csvs", file_name)

        parent_file.extractall("datasets/" + dataset_folder)

        root_folder = os.path.join(settings.MEDIA_ROOT, dataset_folder)
        all_files = glob.glob(os.path.join(root_folder, "*.csv"))

        output_file = os.path.join(root_folder, "output.csv")

        first_one = True
        for csv_file_name in all_files:

            if not first_one: # if it is not the first csv file then skip the header row (row 0) of that file
                skip_row = [0]
            else:
                skip_row = []

            chunk_container = pd.read_csv(csv_file_name, sep=',', chunksize=CHUNK_SIZE, skiprows = skip_row)
            for chunk in chunk_container:
                chunk.to_csv(output_file, mode="a", index=False)
            first_one = False

        for csv_file_name in all_files:
            os.remove(csv_file_name)

        opened_file = open(output_file, "r")
        file_instanse = File(opened_file)
        datafile.csv_file.save(file_name + ".csv", file_instanse, save=True)

        opened_file.close()

        os.remove(output_file)

    return redirect('home')

def create_hdf5(request, datafile_id):

    datafile = DataFile.objects.get(id=datafile_id)
    dataset = datafile.dataset

    zip_list = [datafile.file]

    for file in tqdm(zip_list, leave=False, desc='Converting to hdf5...'):
        # Setting up the files, and directories
        zip_file = zipfile.ZipFile(file)
```

```
        output_file = file.name.split('\\')[-1][:-3]+'hdf5'
        output = os.path.join(settings.MEDIA_ROOT, output_file)

        # Check if a converted file already exists: if it does skip it, otherwise read in the raw csv and convert it
        if (os.path.exists(output) and os.path.isfile(output)):
            pass
        else:
            # Importing the data into pandas
            pandas_df = [pd.read_csv(zip_file.open(text_file.filename),
                                     encoding='utf-8',)
                         for text_file in zip_file.infolist()
                         if text_file.filename.endswith('.csv')][0]
            # Rename some columns to match the more well known dataset from
            # http://stat-computing.org/dataexpo/2009/the-data.html

#pandas_df.rename(columns=rename_dict, inplace=True)

            # Importing the data from pandas to vaex
            vaex_df = vaex.from_pandas(pandas_df, copy_index=False)

            # Export the data with vaex to hdf5
            vaex_df.export_hdf5(path=output, progress=False)

            opened_file = open(output, "rb")
            file_instanse = File(opened_file)
            datafile.hdf5_file.save(output_file, file_instanse, save=True)
        return redirect('home')

def create_dataset_hdf5(request, dataset_id):
    dataset = DataSet.objects.get(id=dataset_id)
    dataset_files = DataFile.objects.filter(dataset_id=dataset_id)

    dataset_hdf5_files = list()

    for file in dataset_files:

dataset_hdf5_files.append(file.hdf5_file.path)

    master_df = vaex.open_many(dataset_hdf5_files)
    output_file = dataset.dataset_name + '.hd5'
    output = os.path.join(settings.MEDIA_ROOT, output_file)
    master_df.export_hdf5(path=output, progress=True)

    opened_file = open(output, "rb")
    file_instanse = File(opened_file)

    data_hdf5_file = DataSetHdf5File.objects.create(dataset=dataset)
    data_hdf5_file.hdf5_file.save(output_file, file_instanse, save=True)

    return redirect('home')

def get_total_csv(request, dataset_id):
    dataset = DataSet.objects.get(id=dataset_id)
    datafiles_zip = DataFile.objects.filter(dataset=dataset)
```

```
    CHUNK_SIZE = 50000

    first_one = True
    output_file = os.path.join(settings.MEDIA_ROOT, dataset.dataset_name + "_output.csv")

    count_of_no_csv = 0;

    for csv_file_name in datafiles_zip:
        if csv_file_name.csv_file == "":
            count_of_no_csv = count_of_no_csv + 1
            continue

        if not first_one: # if it is not the first csv file then skip the header row (row 0) of that file
            skip_row = [0]
        else:
            skip_row = []

        chunk_container = pd.read_csv(csv_file_name.csv_file, sep=',', chunksize=CHUNK_SIZE, skiprows = skip_row)
        for chunk in chunk_container:
            chunk.to_csv(output_file, mode="a", index=False)
        first_one = False

    if count_of_no_csv == len(datafiles_zip):
        return redirect('home')

    opened_file = open(output_file, "r")
    file_instanse = File(opened_file)

    data_hdf5_file = DataSetHdf5File.objects.get_or_create(dataset= dataset)
    data_hdf5_file.csv_file.save(file_name + ".csv", file_instanse, save=True)

    return redirect('home')

def queries(request):
    datasets = list(DataSet.objects.filter(user_id=request.user.id))

    result_datasets = []

    for d in datasets:
        data_file = DataSetHdf5File.objects.filter(dataset_id=d.id)
        if len(data_file) > 0:
            if data_file.last().hdf5_file != "":
                df = vaex.open(data_file.last().hdf5_file.path)
                d.columns = df.get_column_names()
                result_datasets.append(d)

    return render(
        request,
        'app/queries.html',
        {
            'title' : 'Queries',
            'userDatasets' : result_datasets
        }
    )

def get_global_stat(request, dataset_id):
    start_time = datetime.now()

    datafile = DataSetHdf5File.objects.filter(dataset_id=dataset_id).last()
    df = vaex.open(datafile.hdf5_file.path)
```

```
    p_df = df.describe()
    html_obj = p_df.to_html()

    endTime = datetime.now() - start_time

    return render(
        request,
        'app/query-result.html',
        {
            'title' : 'Query Result',
            'resultData' : html_obj,
            'queryTime' : endTime
        }
    )

def get_column_info(request):
    datasetId = request.GET.get('datasetId', None)
    columnName =
request.GET.get('columnName', None)

    datafile =
DataSetHdf5File.objects.filter(dataset_id=datase
tId).last()

    df = vaex.open(datafile.hdf5_file.path)
    p_df = df[columnName]
    min = df.min(columnName).tolist()
    max = df.max(columnName).tolist()
    mean = df.mean(columnName).tolist()

    html_obj = '''
    <h3>Query Result</h3>
    <table class="table table-bordered table-
dark">
    <thead>
        <tr>
          <th scope="col">Name</th>
          <th scope="col">Row Counts</th>
          <th scope="col">Type</th>
          <th scope="col">Min</th>
          <th scope="col">Max</th>
          <th scope="col">Mean</th>
        </tr>
    </thead>
    <tbody>
        <tr>
          <td>''' + str(p_df.expression) + '''</td>
          <td>''' + str(p_df.shape[0]) + '''</td>
          <td>''' + str(p_df.dtype.name) + '''</td>
          <td>''' + str(min) + '''</td>
          <td>''' + str(max) + '''</td>
          <td>''' + str(mean) + '''</td>
        </tr>
    </tbody>
     '''

    data = {
        'test' : str(p_df.expression),
        'countOfRows' : p_df.shape[0],
        'type' : p_df.dtype.name,
        'min' : min,
        'max' : max,
        'mean' : mean,
        'html_obj' : html_obj
    }

    return JsonResponse(data, safe=False)

def test_query(request):
    datafile = DataSetHdf5File.objects.last()

    df = vaex.open(datafile.hdf5_file.path)
    p_df = df.describe()
    html_obj = p_df.to_html()

    return HttpResponse(html_obj)
```

```
def test_vaex(request):
    df = vaex.example()
    p_df = df.describe()

    html_obj = p_df.to_html()

    return HttpResponse(html_obj)
```

url.py

```
from datetime import datetime
from django.urls import path
from django.urls import re_path
from django.conf import settings
from django.conf.urls.static import
static
from django.contrib import admin
from django.contrib.auth.views import
LoginView, LogoutView
from app import forms, views

urlpatterns = [
    path('', views.index, name='home'),
    path('login/',
         LoginView.as_view
         (

template_name='app/login.html',

authentication_form=forms.BootstrapAuthen
ticationForm,
             extra_context=
             {
                 'title': 'Log in',
                 'year' :
datetime.now().year,
             }
         ),
         name='login'),
    re_path(r'^logout$',
        LogoutView.as_view
        (

template_name='app/loggedoff.html'
        ),
        name='logout'),
    re_path(r'^delete-
file/(?P<datafile_id>[0-9]+)/$',
views.remove_datafile,
name='delete_datafile'),
    re_path(r'^delete/(?P<dataset_id>[0-
9]+)/$', views.remove_dataset,
name='delete_dataset'),

re_path(r'^unzip_dataset/(?P<dataset_id>[
0-9]+)/$', views.unzip_dataset,
name='unzip_dataset'),

re_path(r'^unzip_datafile/(?P<datafile_id
>[0-9]+)/$', views.unzip_datafile,
name='unzip_datafile'),

re_path(r'^create_hdf5/(?P<datafile_id>[0
-9]+)/$', views.create_hdf5,
name='create_hdf5'),

re_path(r'^create_dataset_hdf5/(?P<datase
t_id>[0-9]+)/$',
views.create_dataset_hdf5,
name='create_dataset_hdf5'),

re_path(r'^get_total_csv/(?P<dataset_id>[
0-9]+)/$', views.get_total_csv,
name='get_total_csv'),

re_path(r'^get_global_stat/(?P<dataset_id
>[0-9]+)/$', views.get_global_stat,
name='get_global_stat'),
    re_path(r'^ajax/get_column_info/$',
views.get_column_info,
name='get_column_info'),
    path('test-vaex/', views.test_vaex,
name='test_vaex'),
    path('test-query/', views.test_query,
name='test_query'),
    path('upload/',
         views.upload_files,
        name='upload'),
    path('admin/', admin.site.urls),
    path('queries/', views.queries,
name='queries'),
]+static(settings.MEDIA_URL,

document_root=settings.MEDIA_ROOT)
```

models.py

```
from django.db import models
from django.contrib.auth.models import
User

class DataSet(models.Model):
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
    dataset_name =
models.CharField(max_length=200)
    creation_date =
models.DateTimeField('date created')

class DataFile(models.Model):
    dataset = models.ForeignKey(DataSet,
on_delete=models.CASCADE)
    file = models.FileField(blank = True,
null = True)
    csv_file = models.FileField(blank =
True, null = True)
    hdf5_file = models.FileField(blank =
True, null = True)

class DataSetHdf5File(models.Model):
    dataset = models.ForeignKey(DataSet,
on_delete=models.CASCADE)
```

```
    hdf5_file = models.FileField(blank =
True, null = True)
    csv_file = models.FileField(blank =
True, null = True)
```

forms.py

```
from django import forms
from django.forms import
inlineformset_factory
from django.contrib.auth.forms import
AuthenticationForm
from django.utils.translation import
ugettext_lazy as _

from datetime import datetime, date, time
from .models import DataFile, DataSet

class
BootstrapAuthenticationForm(Authenticatio
nForm):
    """Authentication form which uses
boostrap CSS."""
    username =
forms.CharField(max_length=254,

widget=forms.TextInput({

'class': 'form-control',

'placeholder': 'User name'}))
    password =
forms.CharField(label=_("Password"),

widget=forms.PasswordInput({

'class': 'form-control',

'placeholder':'Password'}))

class FileUploadForm(forms.Form):
    dataset_name =
forms.CharField(label='Dataset Name',
max_length=200);
    file =
forms.FileField(widget=forms.ClearableFil
eInput(attrs={'multiple': True, 'class' :
"form-control-file"}))
```